GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

INTEGRACIÓN DE INTERFACES CEREBRO ORDENADOR INMERSIVAS EN ENTORNOS VIRTUALES

Autor: Pablo Pascual Olivas

Tutor: Yago Sáez Achaerandio

Madrid, Junio 2016

Agradecimientos

Querría dedicar este apartado para agradecer a todas las personas que me han apoyado a lo largo de la realización de este proyecto.

A Yago, por toda la ayuda que me ha ofrecido durante estos meses, todos sus consejos y el tiempo dedicado a leer mis innumerables mensajes, esto le convierte en un tutor inmejorable.

A Alejandro, por la ayuda aportada durante el tiempo de realización del proyecto y por su participación durante las pruebas realizadas del mismo.

A todos los amigos hechos durante estos años de carrera. Sin todas esas experiencias, los ánimos, las risas y la amistad que me habéis ofrecido es muy posible que no hubiese llegado hasta aquí.

Una mención especial a Diego y Arturo por su colaboración a la hora de llevar a cabo pruebas del proyecto, habéis sido de gran ayuda, os debo una buena cena.

A todos vosotros, gracias.

Resumen

El cerebro es el órgano más desconocido del cuerpo humano, por lo que su estudio es altamente valorado entre la comunidad científica. Es debido a esto que en los últimos años han empezado a surgir numerosos estudios relacionados con este órgano y sus funciones más en profundidad.

Las tecnologías de electroencefalografía se encargan, a partir de electrodos, de medir la actividad cerebral de un individuo para su análisis, organizando las señales recibidas en las ondas cerebrales más comunes y relevantes en este tipo de estudios.

Este proyecto se centrará en el estudio de la actividad cerebral en cuanto a pensamientos del individuo para clasificar los mismos utilizando sistemas entrenados previamente. El objetivo final del proyecto será poder realizar dicha clasificación en tiempo real.

Los pensamientos a clasificar se centrarán en pensamientos relacionados con el movimiento de un personaje virtual, la dificultad del proyecto radica en la complejidad del pensamiento debiendo representar a la vez la propia acción, la palabra y la tecla que debería accionar el sujeto para dicho movimiento del personaje.

Palabras clave: Interfaz cerebro-ordenador, Electroencefalografía, Clasificación.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introduction

During this chapter, the most important topics regarding this project will be reviewed, like the main focus of the project and the motivation to complete it. After this first approximation, the dissertation structure of the project will be explained for a better understanding of each chapter for the reader.

1.1. Motivation of the project

All this years, the equipment used in electroencephalography have been purchased mainly by hospitals or research laboratories because of the very expensive price of such equipment. This created a very large empty space referred to investigation of this kind of technologies that could have been filled with the large amount of casual users that develop software in their free time knowing the big number of forums and communities working together in internet.

Nowadays some companies are selling low-cost products related to this kind of technologies, this makes the difference for that type of users, making the approximation to the development of software related to brain interaction much easier.

Emotiv Systems is one of the companies selling this sort of low-cost electroencephalography equipment, their main product, Epoc+ is sold for about 600 euros while the usual price for this set of product is around several thousand euros, this is a really huge difference regarding prices. Obviously the quality of the product is not as good in Emotiv equipment but several studies had proven the value of such technology as it presents good enough results for the number of electrodes used, its wireless connection and the price of the product.

Now that is possible for the casual user to buy such equipment, a new market is being created, so software developed for the use of this type of equipment could be very handy in a short period of time, this will be the main topic of this project.

1.2. Objetives

The main idea of the project is to control the movement of a video game character using mainly brain waves, this will be performed using an electroencephalography headset. This headset will have 16 electrodes positioned using the standardized 10-20 system. Those electrodes will capture 14 brain waves.

A wireless connection between the headset and the computer running the software for capturing the information will be made before the beginning of the data capture session. That data will be saved in a text file for it to be operated later.

Once we have the data, we must pre-process it in the way we want and after we must prepare the file and convert to .arff so the software for classification we will use, called Weka, is able to use the data.

The program created will be able to store data into a prepared to use .arff file, train the desired classification and classify in real time the thoughts of a user.

The main objetives of the project are:

- Develop a program capable of record data in real time in a .txt file using the equipment provided by Emotiv, Epoc+.
- Modify the data stored in the .txt file depending on the type of pre-processing we want to perform.
- Once the pre-processing is finished, modify the file so it can be used in the classification software Weka for study the best way of classification and prepare the basis for the real time classification.
- Prepare the program so that it can create a classifier, train it with a pre-recorded training set of data, record data, pre-process that data and classify the final stream in real time.
- When the classification is performed, send signals to the computer so that it interprets the key it should press in the keyboard, this will automatically send the information to the video game that will move the character.
- Adjust the movement of the character in the video game so that the user has the most immersive experience possible.

1.3. Contents of the project

During this section all the chapters of the projects will be explained briefly to create a simple guide for the reader. The document is divided into nine chapters, also several appendices are added at the end.

- Chapter 1. An introduction to the project is presented, regarding the motivation of the project. The main focus of the project and the objectives are discussed. Also is briefly explained the method selected to archive such objectives.

- Chapter 2. In this chapter the state of art surrounding the project is written. It starts explaining general topics about electroencephalography. It is explained from a general point of view the software and hardware of low-cost Brain Computer Interfaces. After this, the same information is given for the equipment used during the project, Epoc+. Also some articles regarding these technologies are shown.

- Chapter 3. During this chapter the development and the implementation of the project is explained. It starts talking about why the software we are using and the programming language are the most useful for the project. Also all the software and libraries used are briefly explained. The functionality of every class developed for the program is explained given some examples of the code used.

- Chapter 4. Explains very briefly how the process for acquiring the data from the headset works, the starting point and every step until the data set is successfully stored in the computer for its posterior use.

- Chapter 5. Contains a detailed explanation of which kinds of pre-processings for the data have been selected for the project and how are performed. Also in each section of the chapter is mentioned how the data will be affected by this sort of modification.

- Chapter 6. In this chapter the relevant experiments for the project are shown, those are related with the training of a classifier and the posterior classification of another set of data. It includes lots of different tests using several classifiers, sets of data and subjects. The results are given in percentage of correctly classified instances, average precision and F1 score. Also some explanations of the behavior for the results are given.

- Chapter 7. During this chapter there will be given an easy to read version of the result with short explanations for the reader to be able to detect the best performances of the classifiers and sets of data.

- Chapter 8. The planning for the time used for the project, divided for each goal for a better final performance during the period of work. Regulatory framework rules regarding software are listed in case of future develops of the project. Finally the economic framework around the project is explained.

- Chapter 9. In this chapter the reader will find the conclusions of the project regarding the results giving the final evaluation of the global project. Includes future develop points for the project in the way of modifications or improvements.

At the end of the document some appendices are attached. The reader will find an extended summary on the first append, deeper explanations regarding all the process of electroencephalography, how the brain is divided, how neurons and brain waves works during the second append, finally, how the electric signals are captured using the electrodes in the headset during the third append.

Capítulo 2

Estado del arte

2.1. Tecnologías de electroencefalografía

Las tecnologías que captan la "imagen" del cerebro como la electroencefalografía (EEG) han estado hasta ahora en hospitales o laboratorios especializados. Pero ahora las EEG están disponibles a bajo precio usándolas junto a un ordenador, esto está dando lugar a numerosas investigaciones en este campo. Las EEG miden los impulsos eléctricos a traves de los sensores colocados en la superficie de la cabeza de una persona. Estos aparatos son wireless, fáciles de colocar y pueden interactuar con aplicaciones de ordenador.

Se ha desarrollado software capaz de traducir datos EEG en parámetros emocionales o estados de ánimo, intentando entender las respuestas humanas a diferentes situaciones, lugares, imágenes...

Las ondas cerebrales se miden con respecto a su amplitud, entre 10 y 100 micro voltios normalmente, y su frecuencia, entre 1 y 80 Hz. Las ondas EEG se clasifican en diferentes bandas de frecuencia: alpha, beta, gamma, delta, theta y mu. Estas bandas de frecuencia son independientes entre ellas y suelen ser mapeadas en estados mentales específicos. Por ejemplo, valores elevados de la banda alpha (8-12 Hz) indican un estado mental relajado, por otro lado, la banda beta (12-30Hz) predomina durante un estado activo o alerta de la mente, esto puede representar los estados de ánimo de una persona.

Una interfaz cerebro ordenador (BCI) como Epoc+ puede traducir los datos EEG en mensajes específicos o comandos para permitir al usuario comunicarse con por ejemplo un ordenador, su teléfono movil, incluso dibujar en CAD [25].

2.2. Interfaces cerebro-ordenador

En este apartado de la memoria explicaremos lo que son las interfaces cerebro-ordenador, en inglés conocidas como Brain-Computer Interfaces, y como funcionan tanto a nivel de hardware como de software, centrándonos posteriormente en Epoc+ ya que es el material con el que se ha realizado el proyecto.

La definición de interfaz cerebro-ordenador sería un equipo que permita la unión entre un ser humano y un ordenador detectando los impulsos eléctricos del cerebro del humano, comprenderlos y, a través de un programa, traducirlos al lenguaje del ordenador. Esto significaría literalmente interpretar el pensamiento de un ser humano para realizar una acción con una entidad externa a él ya sea de forma digital o física, poniendo como ejemplos sencillos de ambos, mover a un personaje en un videojuego o mover una silla de ruedas. Gracias a que actualmente empiezan a aparecer dispositivos de este tipo de precio reducido, es posible que más programadores e investigadores se sumen al desarrollo de aplicaciones para este tipo de tecnologías.

2.2.1. Software de una BCI

El software que presenta la gran mayoría de estos equipos consta de tres partes principales por lo menos, teniendo en cuenta nuevamente que cada empresa desarrollará las aplicaciones que considere según los objetivos de su producto, estas partes son:

- Configuración:
  Es indispensable en cualquier aparato tecnológico poder configurar el equipo de tal manera que dependiendo de la situación los resultados obtenidos sean los óptimos para la persona que los esté recogiendo. Las funcionalidades básicas de esta parte del software deben ser al menos poder comprobar el estado de los electrodos y si cumplen con las características esperadas en el momento actual, es importante poder determinar también si el electrodo está bien colocado o no antes de empezar a tomar los datos.

- Entrenamiento:
  Durante la fase de entrenamiento, el usuario irá añadiendo datos al software de manera que este podrá ir interpretando cada vez de forma más eficiente los pensamientos del sujeto. Existe la posibilidad de que si el software tiene almacenados más de dos usuarios, debido a toda la informacion almacenada por cada uno de ellos individualmente a lo largo del tiempo, el sistema sea incluso capaz de detectar cual de los diferentes sujetos lleva el equipo puesto en el momento actual.

- Observación de los datos:
  Es indispensable para ver las variaciones en las señales poder ver los cambios en las mismas en tiempo real, esta característica está implementada por la mayoría de las empresas que fabrican este tipo de dispositivos y es una de las que más información da a los usuarios.

2.2.2. Hardware de una BCI

Los equipos de electroencefalografía en la mayoría de los casos comparten los mismos componentes en su totalidad, teniendo en cuenta que cada empresa construye el producto según unos objetivos distintos y usando distintos materiales. Las partes en general de este tipo de equipos son las mostradas en la siguiente figura [3].

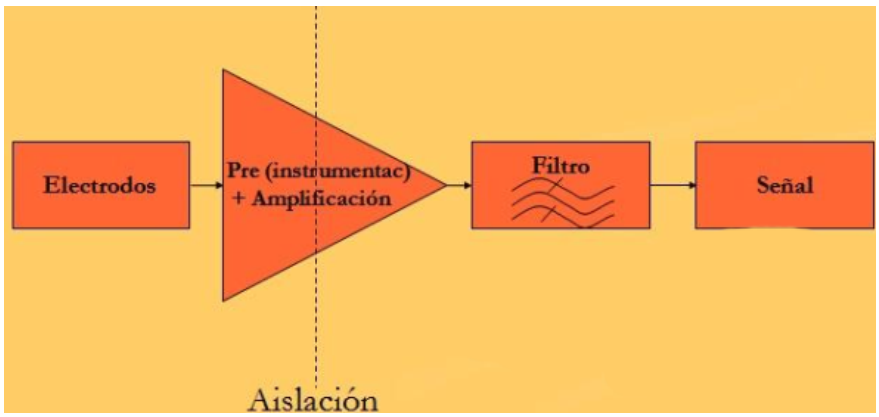

Figura 2.1: Esquema de las partes del sistema de electroencefalografía

- Electrodos:
  Existen dos tipos de divisiones dentro de los electrodos, la primera dependiendo de si están alojados dentro o fuera del sujeto, es decir, si la técnica para posicionarlos es invasiva o no, dado que en la mayoría de los casos en electroencefalografía el método de posicionamiento de los electrodos es fuera del sujeto la división para este tipo de electrodos será la siguiente:

  - Adheridos:
    Su forma es la de platos de metal pequeños que se colocan en el cuero cabelludo del sujeto. Una característica de este tipo de electrodos es su baja impedancia. En la mayoría de los casos estos electrodos se pegan a la cabeza del sujeto por lo que la movilidad del mismo queda fuertemende limitada [7].

Figura 2.2: Fotografía de un electrodo del tipo adherido

  - De contacto:
    Su forma es la de tubos de metal pequeños, en general son de plata. En el extremo por el que se espera recibir las señales eléctricas son colocadas unas almohadillas mojadas en un líquido que ayuda a mejorar la conductividad, el otro extremo del tubo está conectado al dispositivo que captará la señal y la interpretará mostrando el resultado en el equipo.

- Casco de malla:
  Los propios electrodos están posicionados en un casco ya construido, de manera que su colocación es mas rápida y fácil para el sujeto. la forma de los electrodos en este tipo de sistemas es un anillo. Al igual que en los casos anteriores presentarán una impedancia pequeña. Una ventaja importante de este tipo de sistemas es lo poco que pueden afectarle los artefactos generados por equipos externos.

Figura 2.3: Imagen de un casco de malla con electrodos funcionales

- Preparación y amplificación de la señal:
  Durante esta parte del sistema se prepara la señal para interpretar los cambios detectados por los sensores, en nuestro caso los electrodos. Después de este pequeño paso, esa señal será amplificada para conseguir visualizar mejor los cambios y tener una mayor precisión en la representación de la señal. En la mayoría de los casos un equipo de electroencefalografía tendrá una impedancia en la entrada elevada con muy poco ruido y estará bien aislado del sujeto para evitar interferencias en la señal y la toma de datos [11].

- Filtro:
  Al tener tan poco voltaje las señales captadas con un equipo de electroencefalografía cualquier interferencia influye en gran medida nuestros datos, para evitar que esto ocurra, este tipo de equipos necesitan un filtro que elimine ese tipo de interferencias. La mayoría de equipos para captar este tipo de señales eléctricas llevan en su arquitectura un filtro que elimina las frecuencias de la electricidad doméstica, siendo esta de 50 Hercios en Europa y de 60 Hercios en Estados Unidos[29].

- Envío de datos al equipo:
  Cuando los datos están preparados para ser operados es indispensable que lleguen a un dispositivo capaz de almacenarlo, normalmente un ordenador, las formas de transferencia más comunes son:

- Por cable:
  Esta conexión es la más rápida y fiable, simplemente se conecta un cable a ambos extremos, siendo uno la salida del equipo de electroencefalografía y el otro el ordenador donde se almacenarán los datos. Bien es cierto que debido a la distancia con el equipo no es tan necesaria una conexión por cable ya que limita la movilidad del sujeto.

- Wireless:
  La forma más normal de implementer el método wireless en este tipo de equipos es a través de Bluetooth, el equipo de electroencefalografía incorporaría una antena capaz de enviar los datos y el ordenador podría recibirlos simplemente instalándole un software o conectando un USB que reciba los datos.

2.3. Interfaz cerebro ordenador basada en Emotiv Epoc+

Emotiv Systems es una compañía australiana que se dedica a desarrollar interfaces cerebro-ordenador. Es la empresa creadora de Epoc+, una tecnología de encefalografía.

Esta tecnología nos proporciona acceso a datos EGG de alta calidad usando el software para pruebas y el SDK de Emotiv [1].

En este apartado hablaremos más en profundidad tanto del hardware como del software del casco de electroencefalografía Epoc+ ya que es el utilizado durante las pruebas de este proyecto.

2.3.1. Software de Epoc+

Emotiv nos ofrece varias opciones para el software, una versión de pago y una gratuita, durante el proyecto utilizaremos la gratuita llamada "Epoc Control Panel" y se comentarán sus funciones. La versión utilizada será la 2.0.0.21 [2].

- Configuración:
  Nos dará las opciones para afinar el software de tal manera que podamos obtener los resultados óptimos dependiendo de la situación en la que estemos recogiendo los datos, asegurándonos de que todo esté en orden a la hora de comenzar con el procedimiento. Todas las versiones del software vienen con un manual digital de instrucciones que explica el funcionamiento del mismo al usuario para el correcto uso, facilitando así la interactuación con el sistema.

  La manera de conocer si los electrodos están correctamente colocados o no se basa en diferentes colores asociados a la posición de cada electrodo en una imagen que aparece en la pantalla inicial del software. Este color será dado dependiendo de la impedancia media que aparezca en el electrodo, siendo los colores los siguientes:

- Verde: posición óptima del electrodo, significando esto la mejor colocación posible del mismo para la transferencia de datos.
- Amarillo: La posición no es perfecta pero la señal llega al sistema de manera aceptable, de tal forma de que aunque muy probablemente llegue de manera atenuada, la señal será operable.
- Naranja: La posición del electrodo es más bien mala, queriendo esto decir que lo más probable es que la señal que llegue sea errónea o demasiado débil para ser útil en el estudio posterior de los datos.
- Rojo: La señal recibida será demasiado mala para su posterior procesado.
- Negro: sin señal, si este color aparece en alguno de los electrodos, no llegará nada de información al ordenador. En algunos casos, debido al uso alguno de los electrodos o de los contactos se ha podido deformar por lo que quizás cambiando los electrodos de posición se puede arreglar este problema.

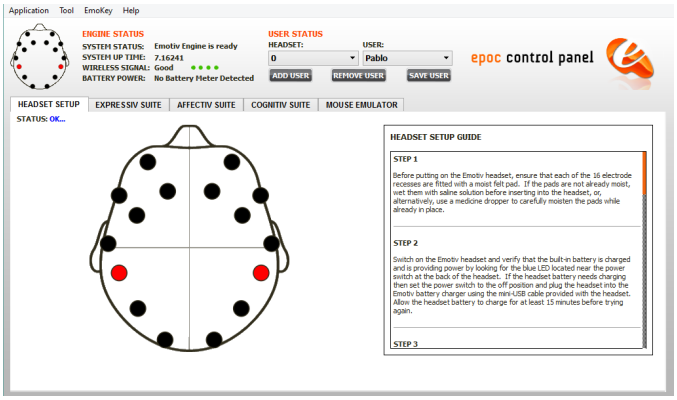

Figura 2.4: Pantalla "Headset setup"

Además de los datos sobre los electrodos podremos ver información como la cantidad de batería restante, la intensidad de la conexión Bluetooth entre el equipo y el ordenador, cuanto tiempo hace que la conexión comenzó entre ambas partes y el usuario actual, permitiendo en este último caso ir cambiando entre usuarios en una pequeña pestaña de selección.

- Expressiv suite:
  En esta pestaña se observa a un personaje virtual animado que imitará en tiempo real los movimientos faciales del usuario, esta es una de las funcionalidades sin necesidad de entrenamiento para el software de este dispositivo, esto es debido a que este tipo de movimientos siguen un patrón genérico entre todas las personas por lo que la señal será reconocida independientemente del usuario.

  A la derecha de esta pestaña se podrán ajustar la sensibilidad de cada acción a las señales recibidas y asignar una tecla física en caso de que el usuario prefiera realizar alguno de los movimientos del personaje a través del teclado.

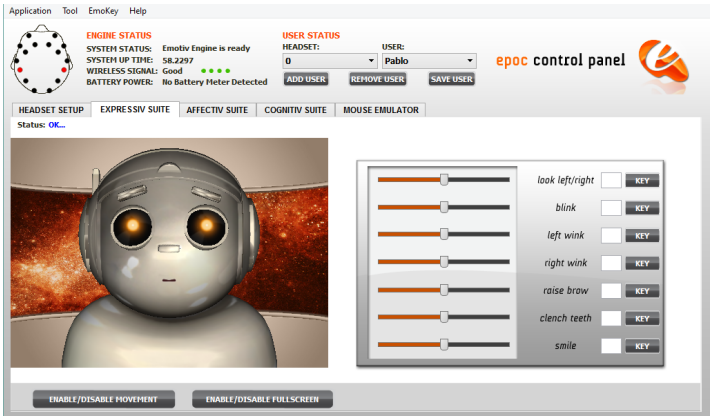

Figura 2.5: Pantalla "Expressiv suite"

- Affectiv suite:
  En esta pestaña el usuario podrá ver una gráfica con varios estados cambiantes relacionados con el sujeto. Entre esos estados se encuentran el estado de calma y excitación del usuario, el estado de concentración y distracción y por último el estado de meditación.

  Además, tendremos esos tres estados en dos versiones diferentes durante el proceso de la toma de datos, la primera versión, mostrada encima, generará los estados en tiempo real, por lo que se podrá ver el estado actual del usuario en lo referente a los tres estados.

  En la segunda versión, mostrada debajo, se verán los tres estados mencionados pero referidos al largo plazo, es decir, durante todo el periodo de tiempo en el que el sujeto tenga puesto el casco, el sistema registrará esa información y la mostrará en esa gráfica para poder ver los cambios en función del tiempo para un periodo largo del mismo.

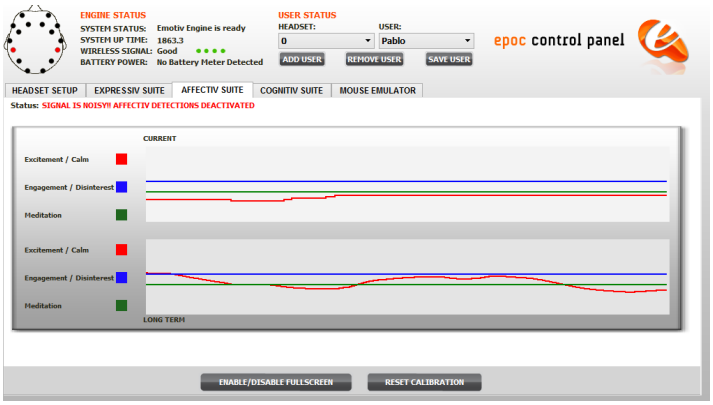

Figura 2.6: Pantalla "Affectiv suite"

- Cognitive suite:
  En esta parte del software se podrá entrenar un clasificador para preparar la selección de diversos pensamientos en tiempo real, los elementos a clasificar serán pensamientos relacionados con el movimiento de un cubo tridimensional.

  Tenemos muchos movimientos como arriba, abajo, izquierda, derecha, traer al frente, mandar al fondo y rotar el cubo. La dificultad de este apartado del software, como en cualquier otro clasificador, radica en el numero de clases a clasificar, cuantas más acciones sean entrenadas, más difícil le resultará al software determinar la acción que el usuario está pensando, tal y como se muestra a la derecha de la pantalla, los niveles de dificultad radican en ese número de acciones, dejando el propio programa un máximo de cuatro acciones posibles aparte del estado "neutral".

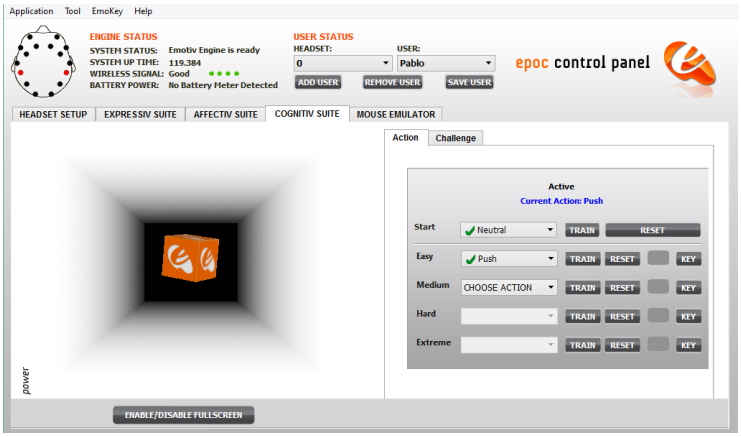

Figura 2.7: Pantalla "Cognitive suite"

- Mouse emulator:
  En esta pestaña se podrá activar un modo de uso del ratón del ordenador a través de los giroscopios del propio equipo Epoc+, con esto se podría mover el cursor por la pantalla simplemente moviendo la cabeza. En la pantalla se muestra una imagen indicando los grados hacia los que avanzaría el cursor en caso de que esta parte del software estuviera activa. En la imagen destaca un circulo que indicaría la dirección actual en la que estamos moviendo el cursor por la inclinación de nuestra cabeza.

  A la derecha están los botones para activar y resetear el software junto con un mensaje informativo para desactivar esta funcionalidad en caso de necesidad desde el teclado. También aparece una barra para ajustar la sensibilidad de movimiento del cursor para la comodidad del usuario.

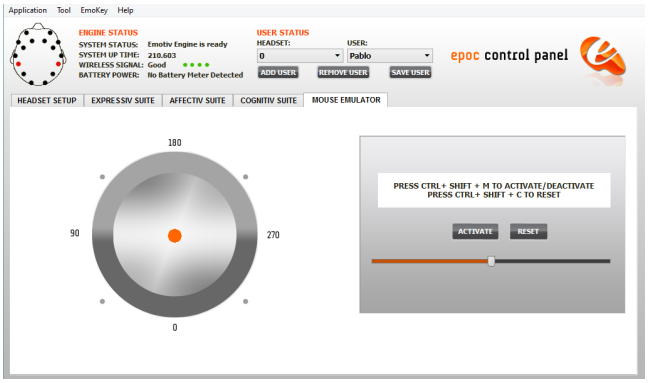

Figura 2.8: Pantalla "Mouse emulator"

2.3.2. Hardware de Epoc+

Tal y como se ha explicado en los conceptos generales de hardware de los sistemas de electroencefalografía, el sistema Epoc+ consta de las partes fundamentales de hardware, el dispositivo completo se puede observar en la siguiente figura.

Figura 2.9: Equipo Emotiv Epoc+

- Electrodos:
  El tipo de los electrodos que usa este sistema es de contacto, es decir utilizarán una almohadilla humedecida con un líquido para facilitar la conducción de la señal eléctrica entre el cuero cabelludo y el metal del electrodo. Una particularidad importante de estos electrodos es que presentan un nivel de impedancia que aumenta en comparación con el instante en el que se aplica el líquido conductor. Por este motivo es importante no usar durante largos periodos de tiempo ya que la variación de la impedancia podría alterar la muestra de datos.

Estos electrodos presentan otro problema común, pasado un tiempo, debido al uso de los mismos, por las condiciones de humedad en los que deben ser usados el metal del que están hechos se acaba oxidando, esto hace que los electrodos puedan acabar perdiendo calidad a la hora de recibir la señal eléctrica.

- Preparación y fitrado:
  Durante esta fase del sistema se obtendrán datos dentro de las frecuencias entre 0.2 y 50 Hercios, se consigue ese rango de frecuencias al filtrar las señales entrantes con dos filtros de quinto orden, uno paso bajo y otro paso alto. Al mismo tiempo, durante el filtrado eliminamos el ruido producido por la red eléctrica. La resolución que obtendremos con este procedimiento será de 51 micro Voltios, manteniendo un ancho de banda de 16 bits. Las muestras por segundo que este dispositivo recogen teoría son 2.048 pero en la realidad tras todo el proceso son 128 debido al coste computacional que representa el uso de los filtros y debido a que los mismos añaden cierta latencia a los datos.

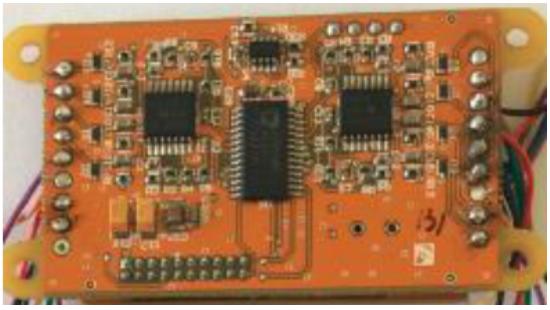

Figura 2.10: Circuitería de la preparación y filtrado de la señal

- Alimentación:
  Este sistema está alimentado por una bateria de Litio, cuyas características principales son 800 mili Amperios, con un voltaje de 3.7 Voltios. Para cargar el equipo se utilizará un cable mini-USB. El propio equipo dispone de un indicador de estado de la batería, mientras el equipo se está cargando el indicador (un diodo LED) aparecerá de color rojo y cuando el equipo esté completamente cargado, aparecerá de color verde. La duración de la batería ronda las doce horas.

- Comunicación:
  El punto fuerte de Epoc+ es este, ya que utiliza la tecnología Bluetooth por lo que la movilidad del sujeto no se ve afectada al utilizar el equipo. Por motivos de seguridad ya que este es un sistema inalámbrico los datos reciben una encriptación antes de ser enviados al ordenador para que una persona no autorizada que interceptase la conexión no pudiese descifrar los datos.

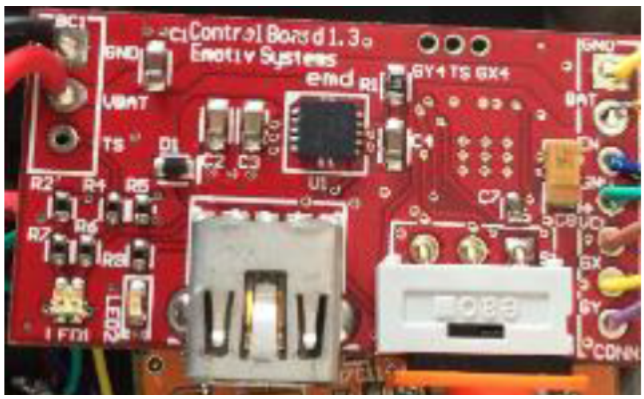

Figura 2.11: Circuitería de la alimentación y comunicación

2.3.3. Emotiv Epoc+

En resumen, las características técnicas de Epoc+ aportadas por el propio fabricante son:

- Señalización:
  Número de canales:
  14 canales (aparte de las referencias CMS/DRL, localizadas en P3/P4)
  Nombres de los canales:
  AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4

- Resolución de la señal:
  Método de sampleo:
  Sampleo secuencial.
  Sampling rate:
  128 SPS o 256 SPS* (2048 Hz internos)
  Resolución:
  14 bits 1 LSB = 0.51 micro Voltios (16 bit ADC, 2 bits ruido instrumental descartado), o 16 bits*
  Ancho de banda:
  0.2 - 43Hz, filtros Notch digitales a 50Hz y 60Hz
  Filtrado:
  Construido con un filtro digital de sincronización de quinto orden.
  Rango dinámico(referenciado a la entrada):
  8400 micro Voltios(pp)

- Conectividad:
  Wireless:
  Banda de 2.4GHz
  Bluetooth Smart

2.4. Enfoques adoptados

ADASTRA

Adastra puede ser utilizado en distintos escenarios, como OpenVibe, que se usa para adquirir y filtrar la información EEG para generar vectores clave a partir de ella. Estos vectores son devueltos a los algoritmos de aprendizaje de Adastra. Este aprendizaje se utiliza para que Adastra pueda detectar acciones como por ejemplo, controlar el cursor del ratón con órdenes como izquierda/derecha, o arriba/abajo.

Este tipo de software puede ser tremendamente útil para personas con diversa discapacidad o dificultad a la hora de usar el ratón del ordenador.

Engaging the brain: Implications of mobile EEG for spatial representation

Inicialmente, los sistemas BCI se desarrollaban con la idea en mente de permitir a las personas con discapacidades neuromusculares a comunicarse o a realizar acciones.

La capacidad de un sistema como Epoc para traducir la actividad cerebral a información útil va ganando importancia como método para entender la experiencia del usuario en un rango de aplicaciones basada en biofeedback y en investigación. Una de las principales ventajas del EEG frete a GSR(respuesta galvánica de la piel) permite a los investigadores identificar las regiones del cerebro que se activan y la parte que toman en el comportamiento.

El sistema Epoc se diseñó para testear videojuegos y para diseñar juegos que respondieran en alguna forma a los estados emocionales del jugador. Su intención es mejorar la estimulación visual y las experiencias por pantalla, como visualizar imágenes, películas, o jugar a videojuegos. También proporciona datos EEG sin tratar, las plataformas de procesamiento de EEG tienen librerías de software para integrar Epoc en estudios sobre neurociencia.

El neuromarkentig (Lee et al) propone que se debería intentar entender la base neural de preferencia, el comportamiento de mercado, las elecciones del consumidor, confianza en marcas, precios, negociaciones o incluso redefinir la ética del marketing.

George Mackerron es el creador de Mappiness, una aplicación de smartphone basada en investigación crowd-sourced. El objetivo de Mappiness es explorar la geografía de las emociones en Reino Unido preguntando a sus usuarios regularmente como se sienten y para proporcionar información sobre su localización, actividad y situación, creando así un estudio geográfico extensivo sobre la experiencia afectiva de un gran número de individuos [23].

Figura 2.12: Dispositivo Epoc Insight

Estimation of Eye Closure Degree Using EEG Sensors and Its Application in Driver Drowsiness Detection

Normalmente , el cálculo del ECD (grado en el que los ojos están cerrados) se basa en procesado de imagen, pero este requiere una iluminación adecuada y una distancia a la cámara estable, por lo que alguna limitación podría ser que el conductor trabajase por la noche o si llevaran gafas.

Un sistema EEG no depende de ninguna de estas cosas, y se puede determinar igualmente el grado en el que los ojos están cerrados independientemente del resto de condiciones [9].

Development of a Mobile EEG-Based Feature Extraction and Classification System for Biometric Authentication, Masters Thesis, Aalborg University Copenhagen, June 2012

Usar ondas cerebrales para autenticar usuarios en sus dispositivos tiene ventajas frente a otro tipo de autenticaciones biométricas como las huellas dactilares o escáneres del iris dado que no es posible que las ondas cerebrales o los pensamientos sean leídos por otras personas [17].

Brain-Computer Interface Based on Generation of Visual Images

Hay trabajos recientes que demuestran la habilidad humana para voluntariamente regular la actividad de las neuronas responsables de la generación de imágenes, pero estos experimentos se basan en métodos invasivos. Datos de MRI reales sugieren que varios patrones de activación del cerebro están correlados con formas específicas de imaginar o de percibir imágenes. De acuerdo con estos datos, la generación de imágenes se activa en la misma zona del cerebro en la que se produce la percepción de imágenes.

Se ha demostrado también que los patrones de actividad del cerebro varían no solo con el tipo de imágenes si no también entre imágenes del mismo tipo, el análisis de esos patrones podría permitirnos identificar la imagen que el sujeto está observando, incluso analizando los patrones de los datos EEG general las imágenes que el cerebro imagina.

Una posible utilidad podría ser crear retratos robot de forma casi instantánea.

Enfocando de otra manera la parte del cerebro que se quiere analizar podría crearse un programa para generación de texto basado en lo que el usuario quiere comunicar [16].

IpsiHand: Direct Recoupling of Intention and Movement (Washington University in St. Louis)

Las lesiones cerebrales traumáticas (TBI) causan pérdidas unilaterales del del control motor a largo plazo debido al daño cerebral en el lado opuesto del cuerpo. Terapias neurológicas convencionales han descubierto la ineficiencia de rehabilitar las funciones de los miembros tras el daño cerebral. Dispositivos BCI toman directamente las señales cerebrales, por lo que parecen prometedores a la hora de proporcionar rehabilitación a estos miembros. El principal problema es que si debido al daño, las señales cerebrales han sido eliminadas, este tipo de sistema no puede funcionar.

Este artículo presenta la IpsiHand. Estudios recientes han demostrado que durante el movimiento de la mano el hemisferio cerebral del mismo lado de la mano también se activa. Usando datos EEG para grabar estas señales y procesarlas podría conseguirse que el hemisferio sano pudiera controlar ambas manos [5].

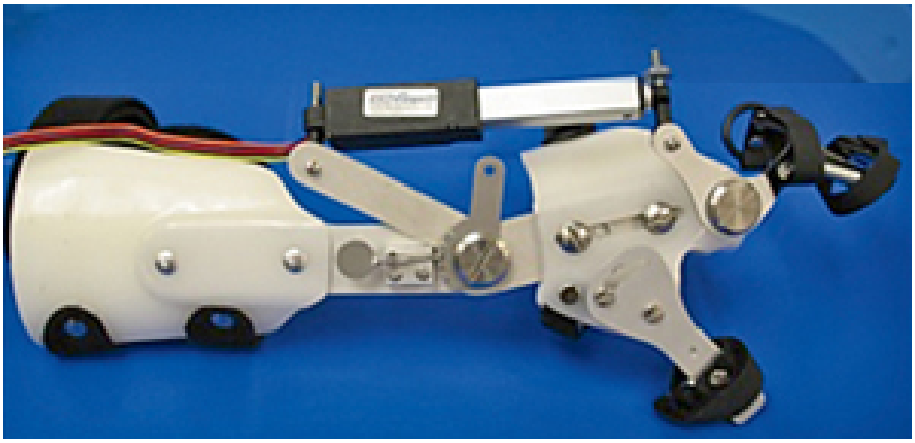

Figura 2.13: Modelo funcional de IpsiHand

NeuroPhone: brain-mobile phone interface using a wireless EEG headset. Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds

Un cambio en la forma de usar los teléfonos móviles sería controlándolos con las señales neuronales del cerebro sin necesidad de usar las manos. Recopilando datos de sistemas EEG sería posible crear aplicaciones que basaran su interacción en este tipo de datos. Utilizando los principios P300 podría ser posible, por ejemplo, llamar al contacto deseado sin necesidad de buscarlo en la lista de contactos [20].

Using brain-computer interfaces to detect human satisfaction in human-robot interaction

Uso de un sistema BCI para obtener feedback emocional del usuario humano como respuesta a la acción de un robot humanoide en entornos de colaboración. El objetivo sería detectar el nivel de satisfacción humano y usarlo como feedback para corregir y mejorar el comportamiento del robot para así maximizar el nivel de satisfacción del usuario [8].

Biofeedback in Virtual Reality Applications and Gaming, University of Massachusetts Lowell. Introduction to Biosensors. Spring 2011

Los videojuegos y la realidad virtual se asocian comunmente a una serie de mandos, joysticks, teclados para el input de información al sistema. Estos sistemas no satisfacen las necesidades de las aplicaciones de realidad virtual que en la actualidad están emergiendo. Las técnicas de biofeedback permitirían al usuario tener un mejor control y tener una mejor inmersión en el mundo virtual a diferencia de los dispositivos actuales.

Los sensores basados en EEG utilizarían las ondas cerebrales del usuario para interactuar con el entorno virtual de formas más naturales que el movimiento físico. Se podría conseguir una respuesta emocional del usuario capaz de crear o cambiar el entorno de juego a través de sensores GSR/HRV. Combinando ambos sensores se podría mejorar enormemente la experiencia del usuario en realidad virtual y aumentar la eficiencia de, por ejemplo, terapia psicológica o tecnología de asistencia basados en la realidad virtual [14].

Capítulo 3

Implementación del proyecto

En este apartado se hablará de la implementación seguida para cada parte del proyecto, todas ellas serían llevadas a cabo utilizando el lenguaje de programación Java y el entorno virtual Eclipse Mars 1. Antes de empezar se explicará el motivo por el que dicho lenguaje de programación, entorno virtual y software de clasificación fueron seleccionados.

3.1. Alternativas de diseño

En esta sección se comentarán las alternativas pensadas para el desarrollo del proyecto, sus ventajas e inconvenientes y se finalizará comentando la decisión del lenguaje de programación y el software seleccionado.

3.1.1. Lenguaje de programación

Al principio del proyecto se pensó en cuatro alternativas para trabajar, estas fueron las siguientes:

- Java:
  Las principales ventajas de este lenguaje eran la fluidez con la que se desenvolvería el programador y el entorno virtual Eclipse, también conocido por el mismo. otras ventajas podían ser la extensa cantidad de librerías disponibles, la facilidad para unirlo con Weka y con el propio software del casco Epoc+, su buena portabilidad y su documentación.

  Sus principales inconvenientes eran la facilidad para escribir un programa que no fuera óptimo, bugs en las librerías y que las clases primitivas no hereden de objetos.

- C:
  Entre las principales ventajas de C encontramos su flexibilidad, limpieza, potencia y su facilidad para ser usado en programas C++.

Los mayores inconvenientes fueron la dificultad para manejar strings y la necesidad de liberar memoria en cada parte del programa.

- Python:
  La principal ventaja de python fue la facilidad de lectura y escritura haciendo que el código sea mucho más corto al final.

  Los inconvenientes pensados fueron la lentitud de python comparado con otros lenguajes y el menor número de usuarios que lo utiliza en caso de futuras mejoras del proyecto.

- Matlab:
  Este lenguaje fue planteado por el conocimiento que el programador poseía del mismo, este fue su principal y única ventaja.

  Fué rápidamente descartado al verse la poca facilidad a la hora de unirlo con las otras partes del proyecto distintas a la clasificación.

Finalmente se decantó la balanza a favor de Java dado que es el lenguaje con el que el programador se desenvolvía con más soltura. Se tuvo en cuenta que el desarrollador también conocía el funcionamiento del entorno virtual Eclipse por lo que el trabajo se facilitaría aún más.

3.1.2. Software de clasificación

Debido a que el desarrollador había aprendido a usar recientemente la herramienta de clasificación Weka, hubo pocas dudas en cuanto a este aspecto del proyecto por el conocimiento poseído al inicio del proyecto sobre este software.

Dicho programa también contiene un número muy elevado de clasificadores distintos con los que hacer pruebas para sacar diversas conclusiones durante todo el proceso.

De manera adicional se utilizó en una de las pruebas el software Sklearn debido a que el software Weka no posee un clasificador concreto que ofrecía unos resultados muy prometedores, el clasificador “Extra Trees”.

3.1.3. Selección de librerías

Se planteó la idea al principio de desarrollar librerías para la toma de datos desde el casco pero se vió que era un objetivo inviable para un Trabajo de Fin de Grado por lo que finalmente se decidió utilizar las librerías del IEDK de Emotiv Systems para la toma de datos. Estas librerías requerian a su vez de las librerías JNA y JRE de Java.

Ya que se utilizaron el sotware de Weka y Sklearn las librerías de ambos fueron necesarias para la implementación del proyecto.

3.2. Librerías

Antes de hablar de la implementación individual de cada parte del programa y de los distintos pasos que ha seguido el proceso durante el proyecto hay que tener en cuenta que desde el principio han sido necesarias librerías para poder utilizar el conjunto completo del programa.

Para empezar, han sido necesarias para la utilización del equipo Emotiv Epoc+, así como parte del software de la mísma compañía, dado que sin estas librerías las funcionalidades como la obtencion de datos en bruto del casco a partir de los electrodos están limitadas o sufren de encriptación.

Las librerías Utilizadas con respecto a esta parte del proyecto se denominan Iedk y constan de las partes mostradas en la siguiente figura.

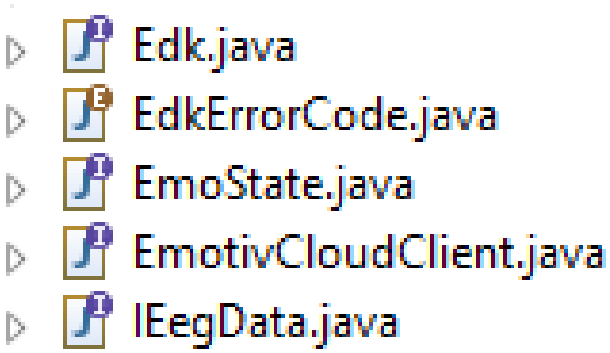

Figura 3.1: Librerías contenidas en el Iedk de Epoc

Junto a esas, han sido necesarias librerías tanto del JRE como del JNA de Java, dado que algunas de las funciones llaman a otras contenidas en dichas librerías. Las versiones utilizadas serán la version JavaSE-1.8 del JRE y la versión 3.0.9 del JNA.

Una vez tengamos las librerías mencionadas hasta este punto será posible capturar datos del casco Epoc+ de Emotiv Systems con la programación adecuada.

Para finalizar, dado que durante el proceso del programa se realiza el entrenamiento del clasificador y clasificación de los datos que se tomen en tiempo real, necesitaremos las librerías referentes al software de clasificación de Weka, que será el software que usaremos para dicha tarea.

Estas librerías estarán preparadas para la versión de software 3.7.3 de Weka.

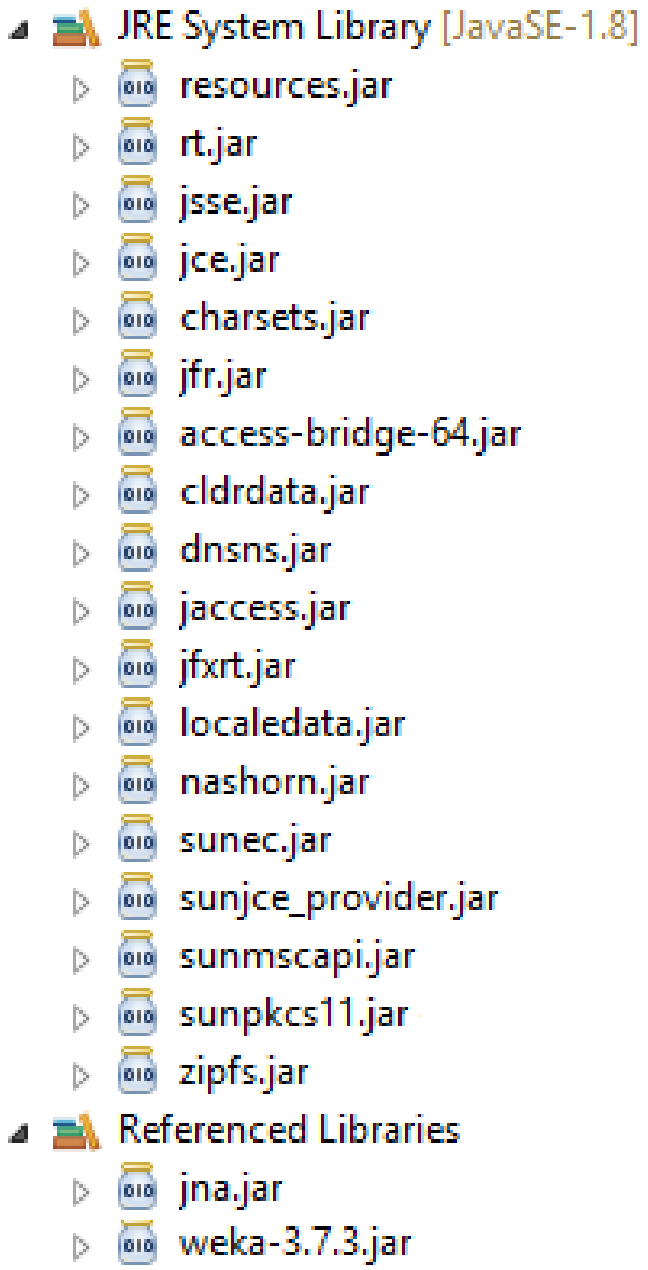

Figura 3.2: Librerías del JRE y JNA de Java junto con las de Weka

3.3. Funcionamiento de la clase para la toma de datos

El primer paso en el proyecto ha sido la toma de datos, esta se ha realizado modificando en parte el software de Emotiv para centrarnos en la manera de enfocar el problema que a nosotros nos interesa.

Comenzamos con la clase "EEGLogger", al principio de esta clase se realiza una llamada a través de "IEE_EngineRemoteConnect()" para conectar con la clase "EmoComposer". Una vez tengamos la conexión, la consola nos avisará.

Después de que la conexión queda abierta, es necesario crear un handler para manejar los datos que se pretende recoger, la variable usada para ello será hData y se creará llamando a la función "IEE_DataCreate()" alojada en "IEegData.INSTANCE". Durante todo el tiempo en el que se reciban datos, el EmoEngine mantendrá un buffer de datos con su sampleo medido en segundos, pero antes de ello, se debe crear el buffer llamando a la función "IEE_DataSetBufferSizeInSec()".

Al haber creado la conexión con el EmoComposer al principio del programa, se habrá creado el registro de un usuario permitiendo una vez que se ha completado este paso la adquisición de datos a través de la llamada a "IEE_DataAcquisitionEnable", una vez esta flag esté activa, el equipo comenzará a registrar los datos EEG para el usuadrio y los almacenará en el buffer. Todos los datos que el casco genere aparecerán en la consola.

```
Pointer hData = IEegData.INSTANCE.IEE_DataCreate();
IEegData.INSTANCE.IEE_DataSetBufferSizeInSec(secs);
System.out.print("Buffer size in secs: ");
System.out.println(secs);

System.out.println("Start receiving EEG Data!");
```

Figura 3.3: Creación variable de datos junto avisos por consola

Cuando queramos recuperar los datos del buffer para utilizarlos en los siguientes pasos, se llamará a "IEE_DataUpdateHandle()", de esta manera, todos los datos recopilados desde la última llamada a esta misma función nos serán devueltos.

Si de alguna forma simplemente quisiéramos obtener alguno de los datos que el casco puede ofrecernos, llamaríamos a la función "IEE_DataGet()". De la manera que se ha orientado el proyecto, usaremos este método que será explicado en los siguientes apartados.

```
IEegData.INSTANCE.IEE_DataGet(hData, i, data, nSamplesTaken.getValue());
```

Figura 3.4: Código para la toma de datos individuales

3.4. Toma de datos de entrenamiento

Ahora que conocemos los conceptos de la clase EEGLogger, comenzaremos explicando las modificaciones aplicadas para realizar la toma de datos de entrenamiento para el clasificador.

Antes de comenzar con cualquiera de los pasos descritos antes, generaremos el archivo .txt que tendrá el nombre que el usuario elija, el archivo será creado en el directorio que el usuario desee. Crearemos una variable String con el directorio donde se encontrará ese archivo y se le darán permisos de escritura a través del programa actualmente en proceso.

Debido al idioma de Windows en el ordenador en el que el proyecto ha sido desarrollado, ha sido necesario dentro del propio programa cambiar el sistema de escritura decimal del lenguaje de "," a "." para que el clasificador Weka pudiera interpretar los datos correctamente.

```
String file_name = "C:/Users/pablopo/workspace/community-sdk-master/epoc/train.txt";
writefile dat = new writefile(file_name, true);

Locale.setDefault(new Locale("en", "US"));
```

Figura 3.5: Selección de archivo, permiso de escritura y formato de escritura

Una vez que el archivo ha sido generado en el directorio deseado se prepara el formato adecuado para ser utilizado por Weka, tal y como se muestra en la siguiente figura:

```
@relation key

@attribute AF3 numeric
@attribute F7 numeric
@attribute F3 numeric
@attribute FC5 numeric
@attribute T7 numeric
@attribute P7 numeric
@attribute Pz numeric
@attribute O2 numeric
@attribute P8 numeric
@attribute T8 numeric
@attribute FC6 numeric
@attribute F4 numeric
@attribute F8 numeric
@attribute AF4 numeric
@attribute key {w,s,neutral}

@data
```

Figura 3.6: Formato necesario para el procesado por Weka

Una vez que el archivo está preparado, el software de Emotiv empieza a funcionar, se crea la conexión entre el casco y el ordenador y comienza la recogida de datos. Esta se realiza de forma cíclica durante un periodo de entre cinco y diez segundos, la variación se ha dado con respecto el proyecto ha ido avanzando, las primeras pruebas para familiarizarse con los datos y los resultados fueron de cinco segundos por las ventajas que ofrecían la rapidez y el mejor manejo de los datos debido a su tamaño, pero una vez que el proyecto fue avanzando mayores registros fueron necesarios.

Antes de comenzar la recogida de datos se da un pequeño periodo de espera, cercano a dos segundos, para evitar que el usuario distraído por pulsar el botón de "inicio de registro" pueda generar datos falsos que confundan al clasificador, este periodo está pensado para que el usuario pueda despejar la mente antes de que comience el proceso.

El bucle se basa en recoger los 14 valores útiles referentes a las ondas EEG que ofrece el casco, al comienzo del proyecto se recibían 17 valores, pero como será explicado en la parte de preprocesado, tres de estos valores acabaron siendo irrelevantes y de utilidad, en la mayoría de los casos, negativa para el clasificador. Durante el bucle se toma valor a valor intoduciéndolo individualmente en el archivo .txt deseado seguido de ","para mantener el formato necesario para Weka, una vez los 14 atributos están escritos en la misma línea, se añade al final la referencia a la tecla, siendo esta "w", "s", "a", "d" o "neutral", las cuatro primeras referencias hacen alusión al movimiento de un personaje digital queriendo simbolizar "adelante", "trás", "izquierda" y "derecha" respectivamente, la última referencia se refiere a un estado mental del usuario en el que no pensará ninguna de las anteriores, es decir, ningún movimiento, asumiendo un estado mental relajado del usuario.

Lo normal para crear los archivos de entrenamiento será crear un archivo diferente para cada clase, haciendo que sean mucho mas manejables los datos durante el preprocesado, aunque dado el código usado para el registro de información sería posible tomar todos los datos juntos dentro del mismo archivo.

Figura 3.7: Ejemplo de muestras guardadas en un archivo de texto

3.5. Entrenamiento y clasificación

Una vez tengamos los archivos necesarios para el entrenamiento del clasificador, pasaremos precisamente a la etapa de entrenamiento y clasificación. Esta etapa utiliza básicamente el mismo código que la clase para la recogida de datos de entrenamiento con algunas modificaciones.

En primer lugar, en vez de crear un archivo para escribirlo, únicamente se leerá el archivo con el que se entrena el clasificador, para ello se crea un buffer de lectura que leerá linea a línea el archivo a través de un FileReader, a este buffer se le llamará brtrain. Para que Weka pueda manejar los datos deberá ser creada una variable de instancias que será denominada traindata y a la que se debe asignar un índice para que al realizar las operaciones pertinentes en el entrenamiento y la clasificación, las funciones de Weka no busquen datos fuera de los límites marcados ya que no deberían encontrarlos. El código utilizado para realizar todo esto se encuentra en la siguiente figura.

```
BufferedReader brtrain = null;
brtrain = new BufferedReader(new FileReader("train2.txt"));
Instances       trainNew;
Instances trainData = new Instances(brtrain);
trainData.setClassIndex(trainData.numAttributes() - 1);
brtrain.close();
```

Figura 3.8: Buffer de lectura de datos desde archivo de texto

Al haber leído el archivo con los datos es imprescindible decirle a Weka el método de clasificación que debe usar, por ejemplo, programado para utilizar Random Forest ya que es uno de los clasificadores con mejores resultados durante el proyecto. En este caso, se creará un nuevo clasificador Random Forest con "new RandomForest()", seleccionando los atributos que se consideren necesarios, el resto Weka los seleccionará de serie, en la realización de este proyecto se modifica el número de árboles aumentando la complejidad y la semilla que variará la generación de números aleatorios en el proceso de generado de árboles, esta semilla es la utilizada ya que por el proceso de prueba y error se ha demostrado que es la que mejores resultados da.

Finalmente se genera el clasificador basándose en los datos de entrenamiento haciendo una llamada a "buildClassifier()". El proceso total quedaría como en la siguiente figura.

```
RandomForest rf = new RandomForest();
rf.setNumTrees(700);
rf.setSeed(2);
rf.buildClassifier(trainData);
Evaluation evaluation = new Evaluation(trainData);
```

Figura 3.9: Generado y acondicionamiento del clasificador

Antes de continuar es fundamental generar una variable de evaluación basada en el set de entrenamiento para poder evaluar posteriormente los datos en tiempo real, como se muestra en la última línea de la figura anterior.

Una vez se entra en el bucle de toma de datos los pasos son similares al apartado anterior, la diferencia principal reside en la creación de un archivo .txt y acondicionamiento de la misma forma que antes para que Weka pueda utilizar los datos de evaluación a medida. La creación del archivo se hará de la forma mostrada en al siguiente figura.

```
final File parentDir = new File("C:/Users/pablopo/workspace/epoc");
parentDir.mkdir();
final String hash = "eval";
final String fileName = hash + ".txt";
final File file = new File(parentDir, fileName);
file.createNewFile();
```

Figura 3.10: Creación de archivo .txt

Una vez tengamos el archivo, tendremos que darle derechos de escritura y prepararlo con los atributos necesarios para las operaciones de Weka.

Cuando el archivo está listo, se procede a llenarlo con los 14 atributos pero en lugar de introducir la referencia a la tecla se introducirá "?", que es el símbolo que indicará a Weka que no se posee esa información, de esta manera, se seguirá aceptando esos datos como válidos a la hora de evaluarlos.

Dependiendo de las pruebas que estemos realizando los datos se tomaran de una forma u otra y se preprocesarán o no, por ejemplo intentando clasificar datos en bruto simplemente se guardará en el archivo una línea y se le pasará al clasificador para su evaluación, pero si utilizamos algunos de los preprocesados referentes a la media de Laplace, se guadarían en arrays 10 muestras con los 14 atributos más la variable de referencia "?" y se les aplicaría la media según el proceso requiriese, una vez hecho esto, ese resultado sería el que se escribiría en el archivo .txt.

Cuando el archivo .txt de evaluación tiene los datos deseados introducidos, se crea un nuevo buffer de lectura, una variable de instancias con un índice asignado tal y como se ha explicado anteriormente para el set de entrenamiento y se genera un valor de evaluación con la función "evaluateModelOnceAndRecordPrediction()", a esta función le serán introducidos el clasificador, el set de evaluación y el número de la instancia a clasificar, dado que en nuestro caso la clasificación es en tiempo real solo tendremos una instancia cada vez y este número siempre será cero. Un ejemplo de la línea de código es mostrado en la siguiente figura.

```
double value = evaluation.evaluateModelOnceAndRecordPrediction(rf, evalData.instance(0));
```

Figura 3.11: Línea de código para la evaluación de datos

El valor de la variable devuelta irá desde cero hasta el número total de clases menos uno, indicando las clases en el mismo orden que estén escritas en el atributo "key" del archivo de datos, es decir, tal cual las ha ordenado Weka al leerlas.

Cuando se obtiene este valor, se pasa por una serie de bucles "else if" de tal manera que se entre en el indicado para la tecla correcta. Una vez dentro del correcto el programa le pasa a Windows una señal que representa la orden de pulsación de la tecla buscada con su correspondiente orden de levantamiento de tecla. Este proceso indicará al personaje virtual la direción a la que debe moverse.

Finalmente, dado que no queremos que se repitan muestras en los datos borraremos el archivo totalmente, esto no es problema ya que el coste computacional de crear un archivo .txt en blanco y borrarlo con tan poco peso es minúsculo y por ello la creación del archivo está dentro del bucle.

```
final String file_path = "C:/Users/pablopo/workspace/epoc/eval.txt";
File borrar = new File(file_path);
borrar.delete();
```

Figura 3.12: Borrado del archivo de datos de evaluación

3.6. Media Laplaciana

Varios de los métodos de preprocesado se basarán en la media Laplaciana, calculándola para cierto número de valores, ya sean grupos o el total de ellos.

3.6.1. Media Laplaciana aplicada a grupos de datos

Este será uno de los métodos usados, se basa en la toma de grupos de muestras de diez en diez, y el cálculo de la media Laplaciana para cada una de los 14 atributos, de esta manera se reducirá el ruido, artefactos y el número de muestras reduciendo el tiempo de entrenamiento.

El proceso comienza creando el archivo .txt en el que serán guardados los datos según el preprocesado se vaya realizando, se le darán permisos de escritura tal y como se ha explicado anteriormente.

Será necesario leer del archivo de texto cuyos datos quieran ser preprocesados, por lo que se creará un String con la información del directorio en el que está situado el mismo y su nombre.

Se crea un array de doubles para guardar la línea de datos leída en el instante actual y otro sobre el que se calculará la media. También se crea un contador para leer las líneas de diez en diez.

Se entrará en un bucle de lectura hasta que se llegue al final del archivo tal y como se muestra en la siguiente figura, en el que se dividirá la línea cada vez que el programa lea el símbolo "," de tal manera que guardará en cada posición del array el valor del atributo.

```
try (BufferedReader br = new BufferedReader(new FileReader(file))) {
    String line;
    while ((line = br.readLine()) != null) {
        String[] stringArray = line.split(",");
```

Figura 3.13: Bucle de lectura y separación de valores por símbolo

Se irá sumando en el array el valor de los atributos de la línea leída al valor anterior del array hasta leer las diez líneas, una vez llegados a este punto el valor de cada atributo se dividirá entre diez para obtener el valor medio.

Cuando tenemos el valor medio se escribe el array en el archivo .txt deseado, eliminando el valor de ambos arrays una vez queda escrito en el archivo para evitar errores en la siguiente ronda, además, se escribe nuevamente la referencia a la tecla deseada. Vuelve a empezar el bucle para los siguientes diez valores.

3.6.2. Media Laplaciana aplicada al total de los datos

En este caso se recorrerá dos veces el archivo que se quiere preprocesar. La primera vez para calcular la media de cada atributo, y la segunda para restar la media correspondiente a cada atributo. Esto centrará la media en cero.

Al igual que en el proceso anterior necesitaremos escribir en un archivo .txt, lo crearemos en el directorio deseado con el nombre que el usuario elija y se le darán permisos de escritura nuevamente.

Una vez creado el archivo .txt comenzaremos un bucle de lectura del archivo de datos a preprocesar, este bucle leerá el archivo línea a línea hasta el final del archivo.

Se guardarán los datos leídos en un array de doubles actualizando el valor de las posiciones del array en cada iteración del bucle sumando el nuevo valor leído al valor numérico anterior. Al mismo tiempo se actualizará un contador que se usará para conocer el número total de líneas leídas. El proceso de División de los datos leídos en doubles es el mismo que en el proceso anterior. Una vez se ha leído el archivo completo se divide la suma total de cada atributo del array entre el número de líneas leídas para conseguir la media de cada atributo individual.

Una vez tenemos la media, se vuelve a leer el archivo de tal manera que cada vez que se guarde el valor del atributo en el array se le resta su media. Una vez restada se escribe ese array en el archivo final, añadiendo la referencia a la tecla deseada para terminar cada muestra.

```
double[] re = new double[stringArray.length];
for (int i = 0; i < stringArray.length; i++) {
    String numberAsString = stringArray[i];
    re[i] = Double.parseDouble(numberAsString)-mean[i];
}
```

Figura 3.14: Bucle de centrado en cero de la media de cada atributo

Con este procedimiento no se disminuye el número de muestras, por lo que el tiempo de entrenamiento será el mismo que antes del preprocesado.

3.7. Fast Fourier Transform (FFT) o Transformada de Fourier

Transformando los datos utilizando las ecuaciones matemáticas de la transformada de Fourier se espera dar un nuevo enfoque a los mismos haciendo que el clasificador los interprete de una forma más acertada al ser procesados desde el punto de vista de la frecuencia.

Con este procedimiento conseguiremos acelerar el entrenamiento debido al downsampling que conlleva en nuestro caso, reduciendo el número de muestras que entrenarán al clasificador.

El programa utilizará librerías de sklearn para crear y entrenar el clasificador "Extra Trees". Al comienzo se crean dos variables numéricas indicando el número de puntos de la FFT, siendo en nuestro caso 128.

Se define una constante que guardará la totalidad de los datos por lo que será necesario leer de los archivos de datos, junto con un array que guardará los datos en los grupos determinados.

La totalidad de los datos entrarán en un bucle para su división en ventanas de 128 muestras, tal y como marcamos al principio del programa.

Una vez realizada la división en ventanas se procede a la división por clases, es decir por las posibles teclas a pulsar o pensamientos realizados por el sujeto.

Una vez tengamos todas las divisiones realizadas se procede a la realización de la FFT de todos los datos. Nos quedaremos con los atributos que consideramos más relevantes, siendo estos la media, la mediana, los valores mínimo y máximo, la desviación típica y los percentiles 25 y 75.

Cada vez que se realice el proceso, todos estos atributos serán añadidos al nuevo archivo con los datos transformados que será utilizado para la clasificación.

Capítulo 4

Recogida de datos

La recogida de datos en este proyecto está basada en los datos que nos ofrece el casco Epoc+. Este casco funciona a partir de 16 electrodos que tienen que ser humedecidos previamente, y que se colocarán en su posición correspondiente del cuero cabelludo del individuo que está generando los datos. Es importante la colocación correcta de estos electrodos para exprimir al máximo la eficiencia de este equipo a la hora de recoger los datos.

Los datos que se recogen son 14 valores numéricos de tipo decimal medidos en micro voltios, junto a otros 8 valores que recogen datos como la calidad de la señal u otros valores que pueden ser de interés. Estos datos el software de Epoc los denomina:

ED_COUNTER, ED_AF3, ED_F7, ED_F3, ED_FC5, ED_T7, ED_P7, ED_O1, ED_O2, ED_P8, ED_T8, ED_FC6, ED_F4, ED_F8, ED_AF4, ED_GYROX, ED_GYROY, ED_TIMESTAMP, ED_FUNC_ID, ED_FUNC_VALUE, ED_MARKER y ED_SYNC_SIGNAL.

4.1. Requerimientos

Para poder realizar la recogida de datos será necesario contar con una correcta configuración del casco. Para conseguir esto Emotiv nos ofrece un software de prueba (Emotiv control panel) que nos muestra una imagen de la cabeza del sujeto con las 16 posiciones de los electrodos y un indicador basado en colores de los electrodos. Este sistema de colores será negro, rojo, naranja, amarillo o verde, siendo el color verde indicativo de una colocación de máxima eficiencia, con color naranja de eficiencia media y de color rojo refiriéndose a una eficiencia baja. Este software también nos muestra la intensidad de la señal Wireless del casco que, obviamente, cuanto mejor sea esta, mejor recibirá el ordenador los datos. El receptor de la señal será un USB que tendrá que estar conectado al ordenador en cada toma de datos que se quiera realizar.

Además del servicio de configuración ("setup") para probar las funcionalidades del casco, permite añadir distintos usuarios y cuenta con pequeñas demos para controlar expresiones faciales, medir elementos emocionales, y realizar movimientos con un cubo en un espacio tridimensional basándose en comandos por pensamiento. Para poder realizar los comandos por pensamiento será necesario un proceso de entrenamiento previo que asocie un pensamiento a cada acción deseada.

4.2. Proceso de recogida de datos

Utilizando Eclipse como entorno virtual y Java como lenguaje de programación, se recogerán los datos que el sistema Epoc+ ofrece. Para obtener los datos del casco, también se hará uso de las librerías de funciones que Emotiv ha desarrollado explícitamente para Java junto a las librerías JRE de Java. Con todo esto se elaborará un código que se debe ejecutar.

Una vez se ejecute este programa, en primera instancia las librerías facilitan al ordenador que entre en contacto con el casco de forma inalámbrica y prepara la comunicación para el envío de datos en "stream". Una vez sea establecida esta conexión, el programa entrará en un bucle infinito de recogida de datos dándole previamente un periodo de espera para que el usuario deje la mente en blanco. Esto último es fundamental para facilitar la recogida de datos que en de otra forma podría devolver datos inutilizables al tener influencia de pensamientos y circunstancias previas que afectan tal y como se ha explicado en apartados anteriores.

El bucle infinito que se ejecuta facilita la posibilidad de tomar medidas de diferentes tamaños y de esta forma hacer diversos tests. Estos tamaños del archivo de datos van marcados por el tiempo que se va a dedicar al entrenamiento del casco, donde a mayor tiempo, mayor tamaño tendrá el conjunto de datos recogido.

Una vez esperado el tiempo deseado para la recogida de datos, se para la ejecución del programa. Ahora se puede contar con un conjunto de datos "en crudo", estos no pueden ser procesados directamente ya que es posible que haya datos que sobren, haya que asignar a cada dato una variable o que haya que normalizar los mismos.

Capítulo 5

Preprocesado de los datos

Una vez se cuenta con los datos descargados, se querrá proceder a analizarlos. Para ello en primer lugar es necesario darles el formato adecuado para su análisis, en este caso eso será en formato .txt ya que se usará la herramienta de software Weka, lo normal sería el formato .arff pero usando Weka desde el propio Eclipse da la posibilidad de utilizar archivos de texto. En este caso se trabajará con muchos sets de datos utilizados para sacar diferentes conclusiones.

5.1. Criba de datos innecesarios

De todos los datos que se recogen solo se utilizarán en el archivo los 14 datos correspondientes directamente a valores de la actividad cerebral proporcionados por los electrodos.

El programa a la hora de preparar el archivo generará el archivo .txt con los 14 valores numéricos incluyendo además un valor extra que indicará la tecla a pulsar, nuestras posibilidades serán:

- Neutral: indicando no pensar ninguna acción.
- w: indicando pensar “adelante”.
- s: indicando pensar “atrás”.
- a: indicando pensar “izquierda”.
- d: indicando pensar “derecha”.

Estos valores se asocian a esas teclas dado que este proyecto está orientado al movimiento del personaje en un videojuego donde al igual que el pensamiento la tecla “w” se asocia con caminar hacia adelante y así con el resto de teclas.

Una vez terminado el proceso tendremos los archivos preparados para su utilización con Weka, tanto en entrenamiento del clasificador deseado como en clasificación del set de datos seleccionado.

5.2. Criba de datos selectiva

En un intento de mejorar los tiempos de procesamiento y posibles resultados obtenidos con el preprocesado basado en la criba de los tres datos considerados inecesarios durante la toma de datos, se dió un paso más en cuanto a este proceso procurando eliminar los posibles atributos referentes a las ondas cerebrales que aportasen menos información al clasificador.

Este proceso se basará en quedarse únicamente con la mitad de los atributos del set de datos, es decir, con siete de los catorce atributos que tenemos en este momento, se probará a utilizar tres modelos distintos siguiendo este sistema, quedarse con los siete primeros atributos, los siete atributos centrales o los siete finales.

Hay que tener en cuenta que debido a la aleatoriedad de la forma de la señal entre individuos e incluso entre sesiones este es un estudio arriesgado y deberán investigarse las distintas opciones individualmente en cada caso.

5.3. Media Laplaciana

Se utilizará una aproximación a la media Laplaciana. Para ello existen diversas para realizarla. Para poder obtener la media Laplaciana, usaremos dos métodos distintos, mediante grupos de datos y mediante la resta de la media total a todos y cada uno de los valores.

5.3.1. Media Laplaciana en grupos de datos

En esta aproximación se cogerán bloques de 10 grupos de datos (un grupo de datos contiene 14 datos, uno por cada onda recibida en este software). A ese bloque de 10 se le calculará la media de cada uno de los 14 valores, haciendo que se tengan un total de 14 valores en lugar de los 140. Esto también ayudará al tiempo requerido para el entrenamiento del clasificador haciendo que este se reduzca notablemente.

Lo que se quiere conseguir mediante este procedimiento es eliminar el ruido de la señal en cada canal. Eliminando así sus interferencias y mejorando así la calidad de los datos a analizar. Esto implica que la clasificación será más sencilla y será más fácil obtener un clasificador mejor con estos datos. También es posible que esto suponga la eliminación de cierta información útil, pero al mismo tiempo se considera que las ocasiones en que ocurra esto serán tan pocas que no afectarán al resultado final de la clasificación.

5.3.2. Media Laplaciana en la totalidad de los datos

Este procedimiento es muy similar al anterior. La diferencia radica en que este se basa en restar la media total de cada uno de los 14 atributos a todos los valores centrando de esta forma la media en 0. Con esto se obtiene una reducción de ruido y de todos los artefactos y consecuencias que el ruido puede generar.

El principal problema que conlleva esta aproximación es que el porcentaje de datos útiles perdidos se incrementa frente al método anterior. Esto implica que esta vez sí que es posible que, en el proceso de la limpieza del ruido, se pierda algún dato útil que al contrario que en caso de la media Laplaciana en grupos de datos, que afecte al resultado final de la clasificación.

Puede ocurrir también que algún estímulo produzca un ascenso o descenso general del voltaje del canal. Una posible solución a este problema, podría ser aumentar la muestra, o incluir un tiempo de espera para el antes y el después del comienzo del estímulo. Además, se le puede añadir como un atributo extra la media del canal, no obstante, por cuestiones de tiempo no se ha llegado a probar.

5.4. Fast Fourier Transform (FFT)

Una vez se cuenta con un set de datos donde se ha hecho una selección reduciendo de los 17 datos iniciales a 14, debido a la clasificación tan pobre que esto genera (similares a lo que una clasificación aleatoria ofrecería), se realiza este otro preprocesado basado en la transformada de Fourier (FFT). Con este preprocesado se pretende abrir una nueva vía que facilite el proceso de clasificación de los mismos ofreciendo más precisión.

El primer intento de llevar a cabo este método fue agrupar en bloques de 10 los datos tomados. Esto es un proceso de "downsampling", ya que se reducen las muestras, donde se obtienen ventajas como que el tiempo de entrenamiento necesario para el clasificador se reduce notablemente.

Con este proceso se pretende aproximar desde otro enfoque la clasificación de los datos ya que desde el dominio de la frecuencia es posible que el clasificador sea capaz de asociar mejor los límites de cada tecla con los pensamientos ofrecidos en el set de entrenamiento. Esto lo ofrece la transformada de Fourier ya que este proceso matemático es el que se utiliza para convertir señales que se encuentran en el dominio del tiempo en el dominio de la frecuencia.

Para poder implementar este preprocesado se utilizará el lenguaje de programación Java en el compilador Eclipse y se hará uso de una clase de Java llamada FFT.java que implementa el proceso matemático a nivel de programación. Para poder ejecutar este proceso se llama a la función FFT donde se le pasará un array de números reales y un array de números imaginarios, siendo los números imaginarios 0 para este caso y el de números reales siendo los grupos de 10 de valores que se quieren analizar.

Con estos datos, el código se ejecuta realizando la transformada y guardando en un nuevo archivo los datos preprocesados. En este caso también se eliminará la parte imaginaria que devuelve, ya que casi todos los valores de esta componente son 0 y se entiende que esto apenas aporta información al set de datos, que aumentaría notablemente el tiempo de clasificación para no variar el resultado, o, en el peor de los casos, empeorarlo.

Una vez tengamos el archivo con los datos transformados podrá ser utilizado en Weka ya que por la propia programación del preprocesado quedará preparado con la estructura necesaria para su total funcionalidad en el software. Los sets de datos, ahora con más atributos podrán ser utilizados para el entrenamiento de clasificadores o para la clasificación utilizando clasificadores ya entrenados.

Este proceso no tuvo mucho éxito en los resultados por lo que al final se modificó al explicado en el capítulo 3.

Capítulo 6

Clasificación de los datos

Tras el preprocesado de los datos, el siguiente paso a seguir es el de clasificar los mismos para obtener un sistema que sea capaz de asociar un pensamiento recogido por el casco a una acción, que en este caso será el equivalente a presionar una tecla.

El objetivo último de este proyecto, sería el poder clasificar los pensamientos en tiempo real. Para llegar a tal fin se ha realizado la mayor parte de las pruebas en modo "offline", esto significa que usando Weka se ha intentado clasificar un set de datos habiendo entrenado con otro set de datos distinto al clasificador.

Para esto se realizarán distintas pruebas, una para cada método explicado del preprocesado y aparte pruebas cómo, por ejemplo, intentar clasificar datos de una sesión tomados con los ojos cerrados o datos de una persona con datos de otra persona distinta.

Para llevar a cabo las pruebas se pueden utilizar muchos y muy distintos clasificadores que sirven para propósitos muy variados. En este caso se utilizará principalmente dos de ellos, los cuales son: clasificador "Random Forest" y clasificador "KNN".

Además, iremos probando cada uno de estos clasificadores para los distintos sets de datos preprocesados de diferentes formas tal y como se ha explicado en el apartado anterior. En primer lugar, con los datos en bruto, seguido de la criba, la criba selectiva, la media Laplaciana y por último la FFT.

6.1. Clasificador "Random forest"

El clasificador "Random Forest" está basado en un conjunto de árboles de decisión, de ahí el nombre de "Forest" que significa bosque en inglés, dónde cada árbol depende de valores aleatorios que se prueban independientemente en cada árbol y donde se mantiene constante la distribución aleatoria [12]. El algoritmo "Random Forest" sigue el siguiente proceso:

- 1: Teniendo N casos de prueba tendremos M número de variables en el clasificador.
- 2: Siendo m la cantidad de variables en la entrada usadas para determinar la decisión en un nodo dado; m tendrá que ser mucho menor que M.
- 3: Se determinará un set de entrenamiento para el árbol y se utilizará el resto de casos de prueba para poder aproximar el error cometido.
- 4: En cada nodo del árbol, m variables serán escogidas aleatoriamente para calcular la decisión. Está será determinada a partir del set de entrenamiento.

Cuando se realice la predicción, el árbol será recorrido de arriba abajo, en el nodo donde se termine la búsqueda, será la predicción que se le asigne al caso. Esto se realiza en todos y cada uno de los árboles del "Random Forest" [15].

Entre las ventajas que ofrece se ha de tener en cuenta que el clasificador "Random Forest" es uno de los algoritmos de aprendizaje más efectivos, tiene un buen funcionamiento para un set de datos grande. Además, es capaz de manejar cientos de atributos al mismo tiempo, es capaz de determinar que atributos son importantes y cuáles no, puede llegar a estimar datos que faltan, localiza también valores poco frecuentes y que son muy distintos al resto [6].

Al mismo tiempo, existe la posibilidad en el lado negativo de que lleve a un sobreajuste de datos con ruido [27], es difícil para los seres humanos interpretar la clasificación que realiza el algoritmo debido a lo poco intuitiva que es [4]. Además, en datos con variables correladas, los grupos de variables pequeños, se tienen más en cuenta frente a los grupos de variables grandes.

6.2. Clasificador "KNN"

El clasificador "KNN" ("K nearest neighbours" o "los k vecinos más cercanos") es un algoritmo de clasificación supervisada, es decir, un sistema de aprendizaje que es en un primer lugar basado en un set de entrenamiento. Este método estima los valores de la función de densidad de probabilidad (fdp). Los datos de entrenamiento, son convertidos a vectores de un espacio multidimensional, donde el número de atributos será equivalente al número de dimensiones de ese espacio. Para la clasificación, lo más normal será usar la distancia Euclídea. En el momento de clasificar, los datos serán tratados en un vector en el espacio generado. La distancia Euclídea entre los vectores ya almacenados y el que se acaba de introducir, es procesada y se eligen los k elementos más próximos. El elemento introducido ahora es clasificado con la clase más repetida entre esos k vectores más cercanos.

Un problema asociado a la utilización de esta clasificación es que atributos irrelevantes pueden llegar a condicionar la clasificación final frente a otros atributos que sí que tienen relevancia, sobre todo en el caso en que los atributos irrelevantes se encuentran más cerca (en términos de distancia Euclídea).

Una forma de intentar eliminar este problema, se le puede asignar un valor de relevancia a la distancia entre los atributos [13].

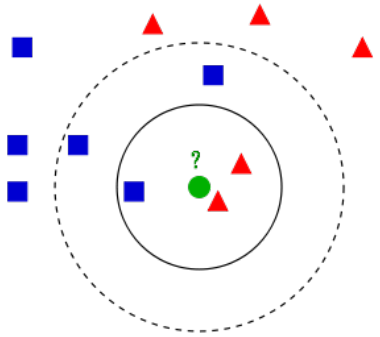

Figura 6.1: Ejemplo de espacio 2D para el clasificador KNN

6.3. Entrenamiento y clasificación con sets de datos de distintas sesiones

Compararando sets de datos de días distintos para un mismo usuario, por lo que guardarían un parecido inferior al óptimo, pero aún mantendrían alguna relación. En este caso, en general, obtenemos resultados algo peores que en los casos en los que los sets han sido tomados durante la misma sesión.

Esta diferencia en los resultados es debida a que la concentración del usuario ha podido cambiar, variando de alguna manera su estado mental, de la misma manera, es casi imposible que los electrodos se mantengan colocados en la misma posición que en la sesión anterior lo que generará diferencias entre ambos datos aún si se les aplicase algún tipo de preprocesado, dificultando así la operación de clasificación para obtener resultados correctos.

6.3.1. Datos en bruto

El primer intento y el más sencillo de clasificar los datos, sería introduciéndolos directamente en Weka (el software encargado de llevar a cabo la clasificación a partir de unos datos insertados), es decir, no realizar ningún tipo de preprocesado en los datos antes de realizar la clasificación. A este proceso se le ha denominado “datos en bruto”.

Para este proceso serán necesarios dos archivos .arff uno de entrenamiento y otro de evaluación. En primer lugar, cargaremos el archivo .arff de entrenamiento que servirá para entrenar el clasificador, como su nombre indica. Una vez entrenado, utilizaremos otro set de datos, denominado set de evaluación, e intentaremos clasificar cada uno de sus datos como una de las clases que hemos identificado en el set de entrenamiento.

Una vez cargado el set de entrenamiento se selecciona el clasificador a entrenar y comienza el proceso de entrenamiento.

En primer lugar, haremos la prueba para un set de entrenamiento y de evaluación con tres clases: “neutral”, “w” y “s”, correspondientes a los pensamientos “neutral”, “adelante” y “atrás” respectivamente.

Tenemos varias pruebas realizadas y cada una con un set de resultados. La primera prueba se basa en utilizar un set de datos grande para clasificar un set de datos pequeño, utilizando los valores que Weka ofrece de manera predeterminada para el clasificador Random Forest. Para estas circunstancias se obtiene un 28,61 % de instancias correctamente clasificadas, frente al 71,39 % que se clasifican incorrectamente, y siendo la precisión media de un 33,4 %. Como se puede observar para el caso con tres variables, estos resultados son muy cercanos al de una selección aleatoria para la clasificación, por lo que nuestro resultado no es fiable. Este es el caso de datos sin preprocesado, es decir, con 17 atributos a analizar por parte del clasificador.

Cuadro 6.1: Matriz de confusión Random Forest predeterminado, datos en bruto

a	b	c	classified as
251	470	529	a=w
229	252	511	b=s
124	133	297	c=neutral

El segundo paso es en el que modificaríamos los parámetros iniciales del Random Forest y no utilizando los valores predeterminados, esto puede hacerse simplemente aumentando el número de árboles. En nuestro caso queda de la forma mostrada en la siguiente figura.

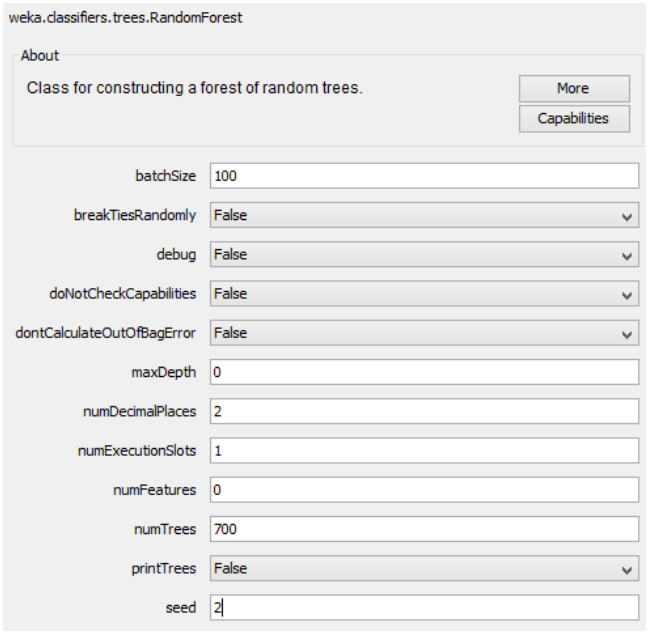

Figura 6.2: Lista de parámetros utilizados en el clasificador Random forest

Al realizar este paso, estamos haciendo que el clasificador Random Forest aumente su grado de complejidad, su capacidad de toma de decisión y, teóricamente, su precisión. Por desgracia, esto no se ve reflejado en los resultados, donde se puede apreciar que las instancias correctamente clasificadas solo han aumentado al 31 % de instancias correctamente clasificadas, frente al 68 % de instancias incorrectamente clasificadas, teniendo en esta clasificación un 36,5 % de precisión media. Esto representa cierta mejora, pero no la esperada frente al cambio en el incremento del coste computacional y del tiempo de espera para el entrenamiento del clasificador. La matriz de confusión en este caso es la siguiente.

Cuadro 6.2: Matriz de confusión Random Forest modificado, datos en bruto

a	b	c	classified as
231	443	576	a=w
216	274	502	b=s
70	122	362	c=neutral

Repitiendo estos mismos pasos para el clasificador KNN, en el primer caso, es decir, con los 17 atributos, obtenemos un 31,5 % de instancias correctamente clasificadas, frente al 68,4 % de incorrectamente clasificadas, teniendo una precisión media del 36,4 %.La matriz de confusión para esta clasificación es la siguiente.

Cuadro 6.3: Matriz de confusión KNN, datos en bruto

a	b	c	classified as
411	393	446	a=w
356	289	347	b=s
50	323	181	c=neutral

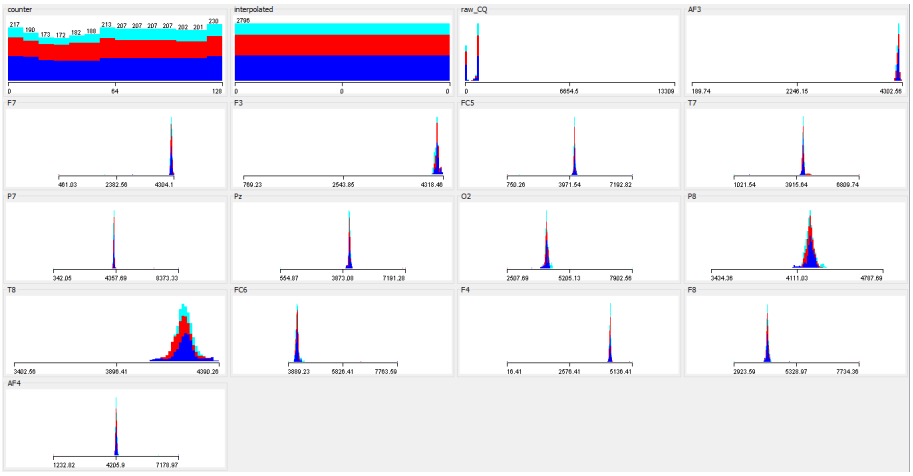

Figura 6.3: Datos del sujeto 1 sin criba visualizados en Weka

6.3.2. Datos cribados en el preprocesado

El siguiente paso a seguir es el de realizar una criba para dejar un total de 14 atributos, eliminando los 3 atributos que hemos considerado innecesarios. Los atributos son "COUNTER", "INTERPOLATED" y "RAW_CQ" que dan datos subsidiarios que o no aportan ninguna información al clasificador o incluso llegan a aportar ruido dificultando al mismo.

Al terminar este proceso nos quedamos con los restantes 14 canales indicando los valores recogidos para las ondas cerebrales, y añadiríamos un atributo extra que en el conjunto de entrenamiento sería la tecla pulsada, y en el conjunto de evaluación se refiere a la variable a estimar.

En este caso, volviendo a utilizar el Random Forest con los valores predeterminados que nos ofrece el software Weka, obtenemos unos resultados del 29,97 % de instancias correctamente clasificadas frente a las 70,03 % incorrectamente clasificadas, teniendo una precisión media del 35,3 %. Como era de esperar este resultado tampoco ha mejorado mucho respecto al intento previo, pese a ello sí que se ha notado una leve mejora al obviar en el set de datos los atributos que solo añadían ruidos a la muestra. La matriz de confusión en este caso es la siguiente.

Cuadro 6.4: Matriz de confusión Random Forest predeterminado, criba

a	b	c	classified as
273	441	536	a=w
223	278	491	b=s
132	135	287	c=neutral

Entrenando al clasificador Random Forest modificando los parámetros a los mostrados en la figura 6.2 obtenemos un 31,01 % frente al 68,99 % incorrectamente clasificadas, todo esto con una precisión media del 36,5 %. La matriz de confusión obtenida es la siguiente.

Cuadro 6.5: Matriz de confusión Random Forest modificado, criba

a	b	c	classified as
251	435	564	a=w
208	281	503	b=s
111	108	335	c=neutral

Como último recurso, hemos intentado utilizar como set de datos de entrenamiento un set muy grande uniendo varios sets. Como resultado de esto, no se ha tenido más remedio que bajar el número de árboles (siendo esto uno de los parámetros que se pueden configurar del clasificador Random Forest) a 400 frente a los 700 que teníamos previamente. Esto trae consigo la consecuencia de una bajada de la precisión en el clasificador. Esto es debido a las limitaciones de los ordenadores que hemos utilizado para el tamaño del set de datos que hemos tratado de manejar.

Realizando este proceso se ha conseguido un 41,13 % de instancias correctamente clasificadas, frente al 58,87 % de instancias incorrectamente clasificadas y teniendo una precisión media de 43 %.

Cuadro 6.6: Matriz de confusión Random Forest modificado, criba, set grande

a	b	c	classified as
1234	1441	267	a=w
1339	2387	211	b=s
1402	1167	451	c=neutral

En el segundo caso, reduciendo los 17 atributos iniciales a 14 en el clasificador KNN (15 si tenemos en cuenta la variable de la tecla), las instancias correctamente clasificadas son 31,3 % frente al 68,6 % incorrectamente clasificadas, y un 38,8 % de precisión media. La matriz de confusión en este caso será la siguiente.

Cuadro 6.7: Matriz de confusión KNN, criba

a	b	c	classified as
425	492	333	a=w
282	330	380	b=s
9	424	121	c=neutral

Las pruebas para el caso restante siempre han empeorado los resultados, así que no las consideramos relevantes para esta memoria. Esto es variando los parámetros del clasificador KNN frente a los parámetros que nos ofrece el software Weka como predeterminados.

Por último, para el caso en el que usamos un set de datos muy grande para entrenar el clasificador KNN, conseguimos un 42,2 % de instancias correctamente clasificadas, frente al 57,8 % de instancias incorrectas y dando una precisión media del 41,2 %. En este caso la matriz de confusión es la siguiente.

Cuadro 6.8: Matriz de confusión KNN, criba, set grande

a	b	c	classified as
723	1393	826	a=w
954	2143	840	b=s
728	980	1312	c=neutral

Podemos observar en los resultados la mejora que representa la eliminación de los atributos innecesarios presentes en los datos iniciales que básicamente aportaban incongruencias al set de entrenamiento haciendo al clasificador confundir clases en un mayor número de ocasiones.

A continuación se muestra una figura del modelo de los datos una vez se realiza la criba de los tres atributos innecesarios para la clasificación.

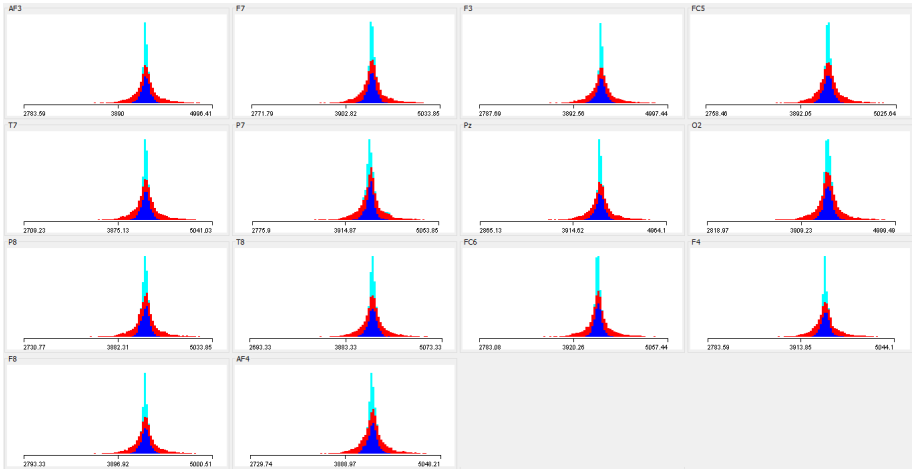

Figura 6.4: Datos del sujeto 1 con criba visualizados en Weka

6.3.3. Criba de datos selectiva

Como prueba para ver si alguno de los atributos de la muestra de datos no aporta información o incluso aporta información negativa, vamos a seleccionar solo parte de los mismos atributos. Esto facilitará la rapidez a la hora de entrenar nuestro clasificador y nos puede servir como ejemplo para futuras pruebas. Las divisiones que haremos será cogiendo la mitad de los atributos de la muestra total, es decir, 7 de los 14 atributos. Los 7 primeros, los 7 centrales y por último los 7 finales.

Los dos últimos casos dieron resultados negativos haciendo que el porcentaje de acierto bajase con respecto al caso anterior. Por tanto, no comentaremos los resultados en este trabajo, centrándonos únicamente en el caso de los primeros siete atributos.

La mejora de la primera mitad puede ser debida a que, con pensamientos complejos como el movimiento, es muy difícil precisar que partes del cerebro responden y cómo lo hacen, y suponiendo variaciones importantes entre una persona y otra.

Durante las pruebas con Weka, los resultados obtenidos utilizando el clasificador Random Forest con los valores predeterminados fueron un 35,4 % de instancias correctamente clasificadas frente al 64,6 % de instancias incorrectamente clasificadas, teniendo una precisión media del 39,6 %.La matriz de confusión es la siguiente.

Cuadro 6.9: Matriz de confusión Random Forest predeterminado, criba selectiva

a	b	c	classified as
449	312	489	a=w
303	190	499	b=s
158	45	351	c=neutral

Utilizando el clasificador Random Forest con los parámetros mostrados en la Figura 6.2 obtuvimos un 35,01 % de instancias correctamente clasificadas frente al 64,99 % de instancias incorrectamente clasificadas, teniendo una precisión media del 39,7 %. En este caso la matriz de confusión es la mostrada a continuación.

Cuadro 6.10: Matriz de confusión Random Forest modificado, criba selectiva

a	b	c	classified as
425	302	523	a=w
297	200	495	b=s
157	43	354	c=neutral

De la misma manera, entrenando con un set muy grande el clasificador Random Forest con los parámetros predeterminados en Weka se obtienen un 35,35 % de instancias correctamente clasificadas, frente al 64,64 % de instancias incorrectamente clasificadas, con una precisión media del 36,1 %. La matriz de confusión en este caso se muestra a continuación.

Cuadro 6.11: Matriz de confusión Random Forest predeterminado, criba selectiva, set grande

a	b	c	classified as
1037	1137	768	a=w
1504	1370	1063	b=s
1057	870	1093	c=neutral

Pasamos ahora al clasificador KNN, seguiremos el procedimiento habitual, primero entrenando con un set único de datos y posteriormente uniremos varios sets para tener un set de entrenamiento grande.

Utilizando el clasificador KNN de Weka con los parámetros de serie obtuvimos un 38,34 % de instancias correctamente clasificadas frente al 61,65 % de instancias incorrectamente clasificadas, teniendo una precisión media del 42,2 %. Obtenemos la siguiente matriz de confusión.

Cuadro 6.12: Matriz de confusión KNN, criba selectiva

a	b	c	classified as
510	269	471	a=w
369	246	377	b=s
194	44	316	c=neutral

Utilizando un set muy grande de entrenamiento para el clasificador KNN se obtiene un 36,49 % de instancias correctas, un 63,51 % de instancias incorrectas con una precisión media del 36,4 %. La matriz de confusión es la siguiente.

Cuadro 6.13: Matriz de confusión KNN, criba selectiva, set grande

a	b	c	classified as
819	1308	815	a=w
1091	1489	1357	b=s
774	941	1305	c=neutral

6.3.4. Media Laplaciana en grupos de datos

Como se ha explicado en la etapa de preprocesado, se cogerán grupos de datos de 10 en 10 y se les aplicará la media. De esta forma, se eliminarán parte de los artefactos que aparezcan en la señal, parte del ruido en baja frecuencia y se reducirá el número de muestras a una décima parte. Los resultados obtenidos son los siguientes:

Con el clasificador Random Forest con los parámetros predeterminados, un 27,95 % de instancias acertadas frente a un 72,05 % con una precisión media del 31,6 %. La matriz de confusión obtenida en este caso es la siguiente.

Cuadro 6.14: Matriz de confusión Random Forest predeterminado, media en grupos

a	b	c	classified as
29	30	66	a=w
32	10	57	b=s
4	12	39	c=neutral

Utilizando el clasificador Random Forest con los parámetros especificados en la figura 6.2, obtenemos un 29,39 % de instancias correctamente clasificadas frente al 70,6 % de instancias incorrectamente clasificadas, con una precisión media del 32,6 %. La matriz de confusión será la siguiente.

Cuadro 6.15: Matriz de confusión Random Forest modificado, media en grupos

a	b	c	classified as
27	32	66	a=w
32	12	55	b=s
4	8	43	c=neutral

Podemos apreciar como nuevamente con el clasificador Random Forest obtenemos unos resultados más bajos de los normales al tener un set de datos de entrenamiento tan pequeño.

Para terminar, utilizamos como set de entrenamiento un set de datos pequeño, de unas 30 muestras para cada tecla. Utilizando el clasificador Random Forest con los parámetros de serie marcados por Weka, obtenemos un 42,65 % de instancias correctamente clasificadas frente al 57,35 % de instancias incorrectamente clasificadas, con una precisión media del 48,6 %. En este caso la matriz de confusión es la siguiente.

Cuadro 6.16: Matriz de confusión Random Forest predeterminado, media en grupos, set pequeño

a	b	c	classified as
52	70	3	a=w
18	40	41	b=s
7	21	27	c=neutral

Si utilizamos los parámetros mostrados en la figura 6.2 para el clasificador Random Forest, obtenemos un 42,65 % de instancias correctamente clasificadas frente a un 57,35 % de instancias incorrectamente clasificadas, todo esto para una precisión media del 49,1 %. Como se puede observar en estos valores, aunque las instancias correctas no aumenten, la precisión media sí que aumenta, esto es un factor que se debe tener en cuenta a la hora de elegir el clasificador mejor preparado para nuestro objetivo. La matriz de confusión se muestra a continuación.

Cuadro 6.17: Matriz de confusión Random Forest modificado, media en grupos, set pequeño

a	b	c	classified as
52	70	3	a=w
18	38	43	b=s
5	21	29	c=neutral

Si utilizamos el clasificador KNN con los parámetros de serie con el que obtenemos un 38,7 % de instancias correctamente clasificadas frente a un 61,3 % de instancias incorrectamente clasificadas, todo esto con una precisión media de un 46 %. La matriz de confusión se muestra a continuación

Cuadro 6.18: Matriz de confusión KNN, media en grupos

a	b	c	classified as
45	54	26	a=w
14	38	47	b=s
11	19	25	c=neutral

El mejor caso hasta el momento para la clasificación de nuestros datos, representando una mejora del 50 % más frente a la selección aleatoria de pensamientos por parte del clasificador se consigue con el procedimiento descrito a continuación.

Reduciendo el set de entrenamiento a 30 muestras nuevamente y utilizando el clasificador KNN de serie, obtendremos un 47,31 % de instancias correctamente clasificadas frente al 52,69 %. La precisión media esta vez será del 55,9 %. La matriz de confusión obtenida durante esta clasificación es la siguiente.

Cuadro 6.19: Matriz de confusión KNN, media en grupos, set pequeño

a	b	c	classified as
53	43	29	a=w
7	60	32	b=s
8	28	19	c=neutral

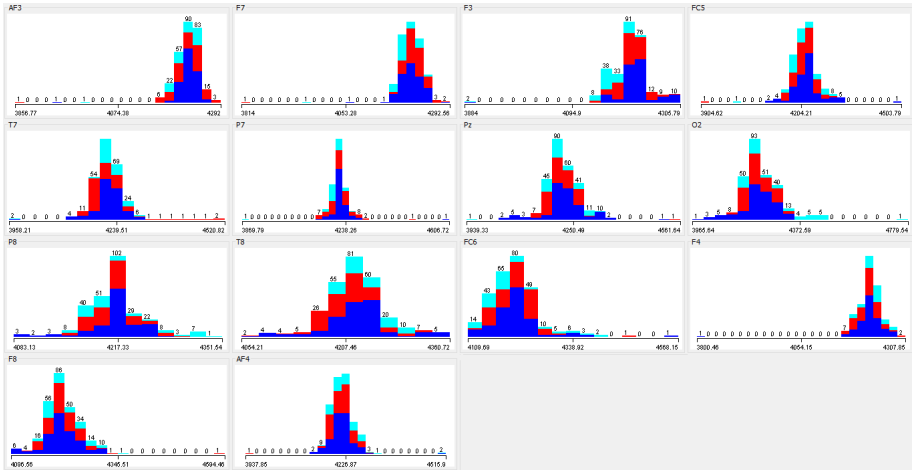

Figura 6.5: Datos preprocesados realizando la media visualizados en Weka

6.3.5. Media Laplaciana en la totalidad de los datos

En este caso restaremos la media total de cada atributo al propio atributo, de esta forma se centrará la media en cero. El objetivo de esta práctica es eliminar en mayor medida el ruido con respecto al anterior tipo de preprocesado basado también en la media Laplaciana, a riesgo de eliminar más cantidad de información útil.

El primer paso del procedimiento es utilizar el clasificador Random Forest con los parámetros de serie, obteniendo un 37,98 % de instancias correctamente clasificadas frente al 62,02 % de instancias incorrectamente clasificadas y con una precisión media del 46 %. La matriz de confusión es la siguiente.

Cuadro 6.20: Matriz de confusión Random Forest predeterminado, resta media

a	b	c	classified as
491	202	556	a=w
252	257	482	b=s
190	50	313	c=neutral

Si ahora se utiliza Random Forest con los parámetros de la figura 6.2 obtenemos un 37,55 % de instancias correctamente clasificadas frente al 62,45 % de instancias incorrectamente clasificadas, todo esto para una precisión media de 46 % .En este caso la matriz de confusión es la siguiente.

Cuadro 6.21: Matriz de confusión Random Forest modificado, resta media

a	b	c	classified as
481	201	567	a=w
242	256	493	b=s
191	50	312	c=neutral

Si reducimos el set de entrenamiento a 30 muestras por tecla y utilizamos el clasificador Random Forest de serie en Weka, obtenemos un 44,25 % de instancias correctamente clasificadas frente a un 55,75 % de instancias incorrectamente clasificadas con una precisión media del 40,5 %. La matriz de confusión obtenida es la siguiente.

Cuadro 6.22: Matriz de confusión Random Forest predeterminado, resta media, set pequeño

a	b	c	classified as
585	553	111	a=w
269	642	80	b=s
203	341	9	c=neutral

Utilizando con el mismo set de muestras pequeño el clasificador Random Forest modificando sus parámetros tal y como se muestra en la figura 6.2 se obtendrán un 45 % de instancias correctamente clasificadas, frente al 56 % de instancias incorrectamente clasificadas, presentando una precisión media del 42,6 %. La matriz de confusión en este caso es la siguiente.

Cuadro 6.23: Matriz de confusión Random Forest modificado, resta media, set pequeño

a	b	c	classified as
621	446	182	a=w
287	602	102	b=s
219	300	34	c=neutral

Hasta el momento, con el clasificador Random Forest, los resultados mediante este procedimiento son siempre mejores que en los casos anteriores en los que utilizabamos la media laplaciana en grupos de diez muestras de datos por lo que parece que este sistema, restando la media total a cada atributo, ofrece resultados prometedores frente al anterior pese al riesgo de eliminar más información útil. Sin embargo, los resultados con respecto al clasificador KNN anterior siguen siendo peores, por lo que a continuación se tendrá en cuenta dicho clasificador utilizando este método de preprocesado.

Utilizando el clasificador KNN con los parámetros predeterminados, obtenemos 40,42 % de instancias correctamente clasificadas frente al 59,58 % de instancias incorrectamente clasificadas con una precisión media del 41,3 %. En este caso la matriz de confusión es la siguiente.

Cuadro 6.24: Matriz de confusión KNN, resta media

a	b	c	classified as
626	381	242	a=w
379	283	329	b=s
236	97	220	c=neutral

Volviendo al mismo set de entrenamiento de 30 muestras por tecla y utilizando el clasificador KNN de serie en Weka, obtenemos un 39,67 % de instancias correctamente clasificadas frente el 60,33 % de instancias incorrectamente clasificadas para una precisión media del 38,1 %. Clasificando de esta manera la matriz de confusión obtenida es la siguiente.

Cuadro 6.25: Matriz de confusión KNN, resta media, set pequeño

a	b	c	classified as
625	499	125	a=w
365	427	199	b=s
296	201	56	c=neutral

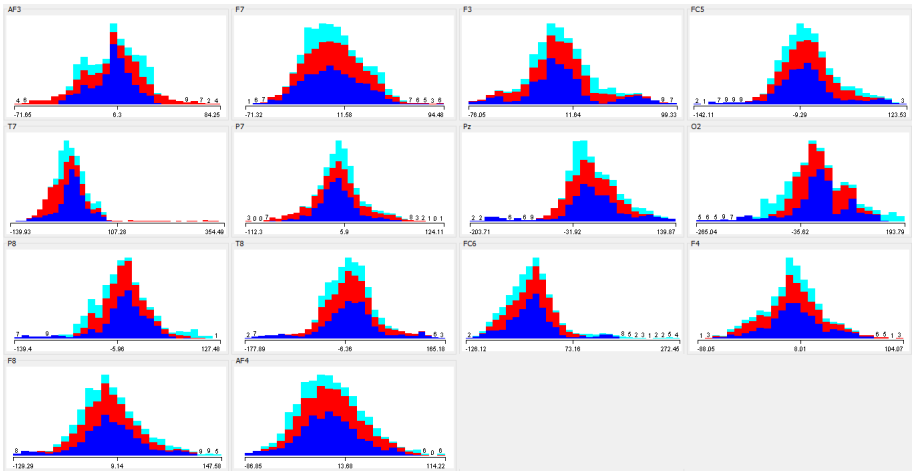

Figura 6.6: Visualización de los datos preprocesados centrando la media en cero

6.3.6. Datos preprocesados mediante la transformada de Fourier

En el caso de los datos preprocesados mediante la transformada de Fourier (FFT) repetiremos los pasos realizados para los casos anteriores.

En el primer caso utilizaremos el clasificador “Random Forest” tal como lo sugiere Weka, sin modificar los valores predeterminados del mismo y utilizando un set de datos de entrenamientos más grande que el set de datos de evaluación.

Los resultados de este procedimiento son del 27,24 % de instancias correctamente clasificadas frente al 72,76 % de instancias incorrectamente clasificadas, teniendo un 32,6 % de precisión media. Como se puede observar, este resultado nos lleva a un porcentaje de acierto mucho menor que los últimos resultados que estábamos obteniendo. La matriz de confusión en este caso se muestra a continuación.

Cuadro 6.26: Matriz de confusión Random Forest predeterminado, FFT

a	b	c	classified as
30	21	4	a=w
54	16	29	b=s
48	47	30	c=neutral

Cambiando los valores predeterminados del clasificador a los valores mostrados en la Figura 6.2, obtenemos un porcentaje de instancias correctamente clasificadas del 25,4 % frente a un 74,5 % de instancias incorrectamente clasificadas, todo para una precisión media del 29,1 %. La matriz de confusión obtenida es la siguiente.

Cuadro 6.27: Matriz de confusión Random Forest modificado,FFT

a	b	c	classified as
21	22	12	a=w
52	12	35	b=s
44	43	38	c=neutral

Los resultados siguen siendo bastante peores que los que hemos llegado a obtener por otros métodos, y peores incluso que los que obtendríamos con un clasificador completamente aleatorio, esto puede deberse a que a usar un número de muestras tan bajo, el clasificador no puede entrenarse correctamente.

Al utilizar el clasificador KNN predeterminado de Weka obtenemos unos resultados de un 34,05 % de instancias correctamente clasificadas frente al 65,9 % de instancias incorrectamente clasificadas, teniendo un 35 % de precisión media. La matriz de confusión es la siguiente.

Cuadro 6.28: Matriz de confusión KNN, FFT

a	b	c	classified as
29	0	26	a=w
55	7	37	b=s
45	21	59	c=neutral

Finalmente vemos que, en cualquier caso, el procedimiento de FFT no mejora los resultados de los datos en bruto para grupos pequeños de sets de entrenamiento, por lo que más adelante se harán pruebas con sets grandes.

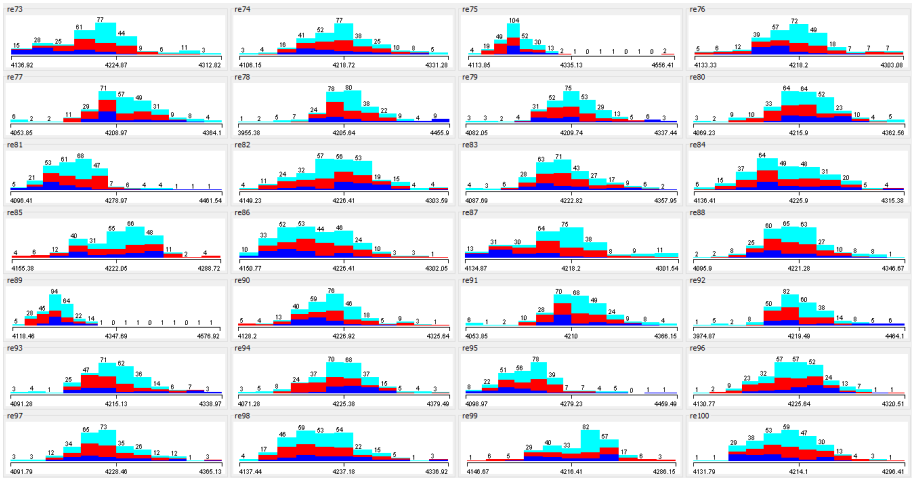

Figura 6.7: Datos tras la tranformada de Fourier visualizados en Weka

6.4. Entrenamiento y clasificación con sets de datos de la misma sesión

Si en lugar de comparar datos de sesiones distintas, realizamos la clasificación comparando un set de datos con un otro de la misma sesión, es decir, ambos sets se han tomado en un corto periodo de tiempo entre pruebas y por tanto manteniendo unas características del entorno muy parecidas. En teoría estos son los datos que mejor resultados nos darán teniendo en cuenta que ambos sets tendrán valores muy parecidos en los atributos para cada una de las clases.

Procurando un entorno libre de distracciones para el usuario como ruidos acústicos y sin mover los electrodos de posición durante la toma completa de datos, se tomaron diez muestras completas de “neutral”, “adelante” y “atrás”, esta fue la toma más larga y aparentemente más limpia del proyecto. Ha tenido mucho valor útil ya que se han conseguido los mejores resultados posibles, igualando los resultados óptimos de Epoc+, pero tiene poca utilidad práctica ya que dicha toma, al igual que para conseguir los mismos resultados con el software de Emotiv, requiere un tiempo demasiado largo para un usuario casual. Alrededor de una hora u hora y media.

6.4.1. Datos cribados en el preprocesado

Ya que se ha demostrado la mejora de los resultados eliminando los atributos “COUNTER”, “INTERPOLATED” y “RAW_CQ” comenzaremos directamente en este paso durante esta parte del proyecto. Asimismo, ya que en la mayoría de los casos con la criba de datos selectiva, dejando únicamente los siete primeros atributos hace que descienda el porcentaje de acierto de la casificación, se obviará ese proceso.

El primer paso, como siempre, es probar en Weka con los datos cribados, se probó a entrenar con un solo set de datos y clasificar otro utilizando el clasificador Random Forest con los ajustes predeterminados de Weka con un solo set de datos y clasificando otro set de datos, los resultados obtenidos son un 61,12 % de instancias correctamente clasificadas, frente al 38,88 % de instancias incorrectamente clasificadas, obteniendo un 68 % de precisión media. La matriz de confusión obtenida es la siguiente.

Cuadro 6.29: Matriz de confusión Random Forest, misma sesión

a	b	c	classified as
1905	533	346	a=w
1466	2954	180	b=s
1369	208	1590	c=neutral

Si entrenamos el clasificador Random Forest cambiando los ajustes a los mostrados en la figura 6.2, los resultados obtenidos mejoran ligeramente, aunque no lo suficiente con respecto al aumento del tiempo de procesado necesario. Los resultados obtenidos serían un 61,32 % de instancias correctamente clasificadas, frente al 38,68 % de instancias incorrectamente clasificadas, presentando una precisión media del 68 %, como podemos observar, este último valor se mantiene igual que en el caso con el clasificador Random Forest predeterminado. La matriz de confusión es la siguiente.

Cuadro 6.30: Matriz de confusión Random Forest modificado, misma sesión

a	b	c	classified as
1881	541	362	a=w
1456	2957	187	b=s
1343	192	1632	c=neutral

Este es el mejor resultado obtenido hasta ahora, por lo que más adelante se investigarán las posibilidades que nos ofrecen otros clasificadores en árbol.

Si entrenamos el clasificador KNN con los ajustes predeterminados de Weka se obtiene un 56,23 % de instancias correctamente clasificadas, frente al 43,74 % de instancias incorrectamente clasificadas, teniendo a su vez, una precisión media del 60,8 %. La matriz de confusión es la siguiente.

Cuadro 6.31: Matriz de confusión KNN, misma sesión

a	b	c	classified as
1438	636	710	a=w
1488	2789	323	b=s
1181	280	1706	c=neutral

Al tener tantos datos el siguiente paso lógico fue crear un set de entrenamiento con la mayoría de los sets de datos, en concreto siete, dado que los dos últimos sets daban unos resultados bastante peores que el resto, seguramente debido al aumento de la impedancia a lo largo del tiempo en los electrodos de este equipo, lo que hace que la señal empeore.

Entrenando el clasificador KNN con los ajustes predeterminados de Weka con esos siete sets y clasificando el set restante se obtiene un 55,59 % de instancias correctamente clasificadas, frente al 44,41 % de instancias incorrectamente clasificadas, teniendo de media una precisión del 54,8 %. La matriz de confusión en este caso es la siguiente.

Cuadro 6.32: Matriz de confusión KNN, misma sesión, gran set de entrenamiento

a	b	c	classified as
1799	1345	1467	a=w
986	2607	280	b=s
1078	382	2527	c=neutral

6.4.2. Media Laplaciana en grupos de datos

Para sets de datos tomados durante la misma sesión repetimos el procedimiento realizando la media en grupos de diez muestras de datos para eliminar ruido e interferencias en la señal, dejando así lo más limpio posible el set de datos además de reducir el tamaño del mismo facilitando el procesado.

Empezamos comparando un set de datos con otro utilizando el clasificador Random Forest con los parámetros predeterminados de Weka. Des esta manera obtenemos un 60,81 % de instancias correctamente clasificadas, frente al 39,19 % de instancias incorrectamente clasificadas, presentando un 65,6 % de precisión media durante la clasificación. La matriz de confusión en este caso será la mostrada a continuación.

Cuadro 6.33: Matriz de confusión Random Forest, media en grupos, misma sesión

a	b	c	classified as
174	56	48	a=w
147	276	37	b=s
103	22	191	c=neutral

Si modificamos los parámetros del clasificados Random Forest a los mostrados en la figura 6.2, los resultados obtenidos son los siguientes, las instancias correctamente clasificadas representan un 61,29 % de la totalidad del set clasificado, frente al 38,71 % de las instancias incorrectamente clasificadas, presentando una precisión media del 65,1 %. La matriz de confusión obtenida es la siguiente.

Cuadro 6.34: Matriz de confusión Random Forest modificado, media en grupos, misma sesión

a	b	c	classified as
166	61	52	a=w
139	283	38	b=s
94	25	197	c=neutral

Si utilizamos Un set grande de entrenamiento y el clasificador Random Forest con los parámetros predeterminados de Weka obtenemos un 56,9 % de instancias correctamente clasificadas, frente al 43,1 % de instancias incorrectamente clasificadas, teniendo una precisión media del 60,1 % en este caso. La matriz de confusión obtenida es la siguiente.

Cuadro 6.35: Matriz de confusión Random Forest, media en grupos, misma sesión, set grande

a	b	c	classified as
203	222	36	a=w
42	337	8	b=s
190	39	169	c=neutral

El siguiente paso a seguir es clasificar un set de datos con otro distinto utilizando el clasificador KNN con los valores predeterminados en Weka. Los resultados obtenidos fueron de un 54,36 % de instancias correctamente clasificadas, frente al 45.64 % de instancias incorrectamente clasificadas, teniendo una precisión media del 57,6 %. La matriz de confusión para este caso es la siguiente.

Cuadro 6.36: Matriz de confusión KNN, media en grupos, misma sesión

a	b	c	classified as
133	78	67	a=w
140	282	38	b=s
116	42	158	c=neutral

Utilizando un set de entrenamiento muy grande uniendo varios sets de datos y clasificando otro set de datos distintos utilizando el clasificados KNN obtenemos un 57,46 % de instancias correctamente clasificadas frente al 42,54 % de instancias incorrectamente clasificadas, teniendo una precisión media del 57,8 % durante la clasificación. En este caso la matriz de confusión obtenida es la siguiente.

Cuadro 6.37: Matriz de confusión KNN, media en grupos, misma sesión, set grande

a	b	c	classified as
165	215	81	a=w
41	335	8	b=s
136	46	216	c=neutral

6.4.3. Media Laplaciana en la totalidad de los datos

Dado que los resultados en el anterior procesado de este tipo de preprocesado fueron peores que los otros resultados obtenidos por otros medios referentes a la media Laplaciana, se obviará este apartado de los datos en el proceso referente a los datos tomados en la misma sesión.

6.4.4. Datos preprocesados mediante la transformada de Fourier

Intentando investigar las mejoras que nos ofrecen los Random Trees, tal y como se ha explicado en la implementación se diseñó un programa con el cual, mediante las librerías de machine learning "sklearn" y el algoritmo de clasificación Extratrees se han obtenido los siguientes resultados utilizando como set de entrenamiento nueve de las muestras tomadas en la misma sesión tras haber sido preprocesadas aplicándolas una tranformada de Fourier de 128 puntos.

Al realizar este procedimiento se utiliza un algoritmo LOSO (Leave One Subject Out), es decir, entrenaremos el clasificador con nueve de los sets de datos e intentaremos clasificar el restante. Los resultados de las diez rondas fueron los siguientes:

- Ronda 1:
  Accuracy: 0.619964
  F1 Score: 0.628754
- Ronda 2:
  Accuracy: 0.731976
  F1 Score: 0.696421
- Ronda 3:
  Accuracy: 0.839849
  F1 Score: 0.843217
- Ronda 4:
  Accuracy: 0.813353
  F1 Score: 0.808162
- Ronda 5:
  Accuracy: 0.757083
  F1 Score: 0.747514
- Ronda 6:
  Accuracy: 0.770821
  F1 Score: 0.757082
- Ronda 7:
  Accuracy: 0.668842
  F1 Score: 0.669062
- Ronda 8:
  Accuracy: 0.664656
  F1 Score: 0.665588
- Ronda 9:
  Accuracy: 0.522985
  F1 Score: 0.499394

- Ronda 10:
  Accuracy: 0.479063
  F1 Score: 0.442834

El programa durante el proceso generará las matrices de confusión de 3x3, tal y como se han visto en el formato de Weka, en de cada una de las rondas. Se pueden visualizar en orden en la siguiente figura.

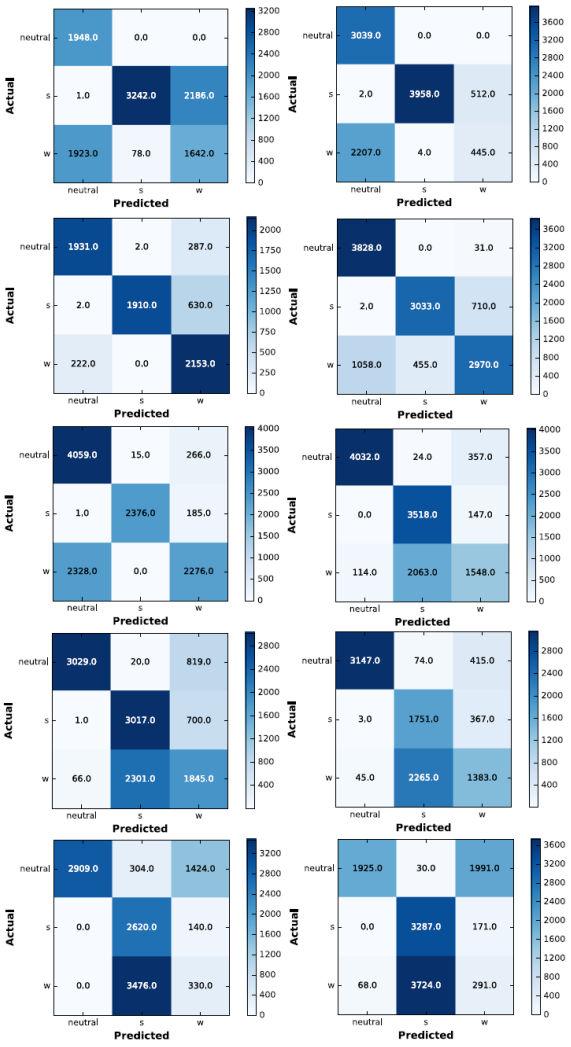

Figura 6.8: Matrices de confusión Extra Trees

Nótense los colores, un azul más oscuro indica más instancias correctamente clasificadas, por lo que el caso óptimo de matriz de confusión sería en la que apareciera color azul oscuro en la diagonal y el resto de casillas blancas.

Podemos ver que por los colores de las matrices de confusión las mejores rondas han sido la tercera y la cuarta, y si nos fijamos en los resultados numéricos así es, de la misma forma los peores resultados se dan en las rondas nueve y diez por lo que repetiremos el experimento descartando ambas rondas, quedando así únicamente con ocho rondas. Los resultados obtenidos son los siguientes:

- Ronda 1:
  Accuracy: 0.839111
  F1 Score: 0.842040
- Ronda 2:
  Accuracy: 0.792171
  F1 Score: 0.774664
- Ronda 3:
  Accuracy: 0.890991
  F1 Score: 0.892871
- Ronda 4:
  Accuracy: 0.895673
  F1 Score: 0.894609
- Ronda 5:
  Accuracy: 0.839910
  F1 Score: 0.840790
- Ronda 6:
  Accuracy: 0.722528
  F1 Score: 0.717718
- Ronda 7:
  Accuracy: 0.634769
  F1 Score: 0.629095
- Ronda 8:
  Accuracy: 0.592275
  F1 Score: 0.601457

Como se observa, los resultados al eliminar esos dos sets mejora notablemente, llegando en el mejor de los casos a un acierto del 89,56 %. Hay que tener en cuenta que este resultado se da en condiciones óptimas pero los mejores resultados de Emotiv también se dan en este tipo de condiciones y con este resultado se ha conseguido igualarlos.

Se debe tener en cuenta que un entrenamiento con ocho sets de datos es un entrenamiento pequeño, cabe la posibilidad de poder incluso mejorar los resultados superando a los de Emotiv si se dedicase más tiempo y más sets de datos a entrenar el clasificador para otorgarle capacidad de decisión en el mayor rango de situaciones posibles, incluso añadiendo sets de sesiones distintas para perfeccionar el sistema al máximo.

6.5. Pruebas adicionales

6.5.1. Entrenamiento y clasificación utilizando un solo set de datos

Hay que tener en cuenta que si usamos las herramientas de Weka de "cross-validation" o de "Percentage Split", utilizando un solo set de datos como set de entrenamiento y clasificación, dividiéndolo para que no aparezcan las mismas muestras en un lado y en otro, los resultados son increíblemente mejores obteniendo para el primer caso (cross-validation) y utilizando "10 folds" un 98,35 % de instancias correctamente clasificadas, frente al 1,64 % de instancias incorrectamente clasificadas, con una precisión media de del 98,4 %.

```
=== Summary ===

Correctly Classified Instances        2750               98.3548 %
Incorrectly Classified Instances        46                1.6452 %
Kappa statistic                          0.9741
Mean absolute error                      0.0694
Root mean squared error                  0.1283
Relative absolute error                 16.3972 %
Root relative squared error             27.8959 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      52.5274 %
Total Number of Instances             2796

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,990    0,019    0,977      0,990    0,983      0,970    0,999     0,999     w
               0,968    0,008    0,986      0,968    0,977      0,964    0,999     0,998     s
               0,998    0,001    0,995      0,998    0,996      0,996    1,000     1,000     neutral
Weighted Avg.  0,984    0,011    0,984      0,984    0,984      0,973    0,999     0,999

=== Confusion Matrix ===

    a    b    c   <-- classified as
 1237   13    0 |    a = w
   29  960    3 |    b = s
    0    1  553 |    c = neutral
```

Figura 6.9: Resultados Random Forest con método cross-validation

En el segundo caso (Percentage Split) usando un valor del 66 % para la división, obtendremos un 98,1 % de instancias correctamente clasificadas, frente al 1,8 % de instancias clasificadas incorrectamente, todo esto para una precisión media del 98 %.

Cabe mencionar que, durante la realización de todas las pruebas, el objetivo principal del proyecto era conseguir mover al personaje de un videojuego en tiempo real, objetivo que alcanzamos. No obstante, el movimiento del personaje era bastante errático debido a que siempre aparecen distracciones del usuario o el set de entrenamiento no cubre todas las posibilidades, pero si tenemos en cuenta el modo offline en el que obtuvimos un 98 % de efectividad al clasificar las instancias, podríamos diseñar un laberinto en el videojuego y hacer que el personaje lo recorriera correctamente, si por ejemplo, el jugador viese el laberinto en vista de pájaro previamente para "entrenar" al personaje buscando que realice las acciones deseadas una vez iniciado el programa.

6.5.2. Entrenamiento y clasificación con sets de datos de tomados de distintos sujetos

Este experimento se basa en comparar los datos tomados de una persona con los datos tomados de otra persona distinta.

En este caso, la precisión es incluso peor que en procentajes aleatorios, ya que la manera de interpretar los pensamientos de un cerebro a otro y la forma que tienen los impulsos de recorrer las redes neuronales varían de forma importante de persona a persona, por lo que resulta imposible tener un buen clasificador a partir del entrenamiento de una persona y que este sea capaz de clasificar correctamente los pensamientos de otra persona.

Hay que tener en cuenta que actividades motoras como los movimientos faciales sí que son comunes entre todas las personas, así que siempre y cuando la persona no tenga lesiones cerebrales, en principio, el casco no tendría problemas para interpretar esas señales.

En un caso de comparativa Entrenando con el set de datos de un sujeto y clasificando un set de datos tomados a otro sujeto, siendo los mismos pensamientos para ambos (estado "neutral", "adelante" y "atrás").

Los resultados obtenidos utilizando el clasificador Random forest con los valores predeterminados son un 27,25 % de instancias correctamente clasificadas, frente al 72,75 % de instancias incorrectamente clasificadas, presentando un 27,1 % de precisión media.

Los resultados Entrenando el clasificador KNN con los valores predeterminados en Weka utilizando el set de datos de un sujeto y clasificando el set de datos de otro resultaron en un porcentaje de acierto del 26,52 % frente al 73,48 % de instancias incorrectamente clasificadas, teniendo de media una precisión del 26,5 %.

Tal y com esperábamos, los resultados obtenidos nos muestran un resultado de acierto realmente bajo, siendo el peor obtenido hasta el momento tal y como esperábamos, esto se debe a las grandísimas diferencias entre los datos de las distintas ondas cerebrales.

Debido a dichos resultados con ambos clasificadores se obviarán el resto de pruebas realizadas dado que presentan una serie de resultados similares con un porcentaje realmente bajo de instancias acertadas durante la clasificación.

Para ver las diferencias entre los pensamientos de ambos sujetos utilizaremos la herramienta de visualización de Weka para comparar la forma que toman los datos de las ondas cerebrales generadas durante los mismos pensamientos. Esta comparación entre visualizaciones se muestra en la siguiente figura, arriba el sujeto uno y abajo el sujeto dos.

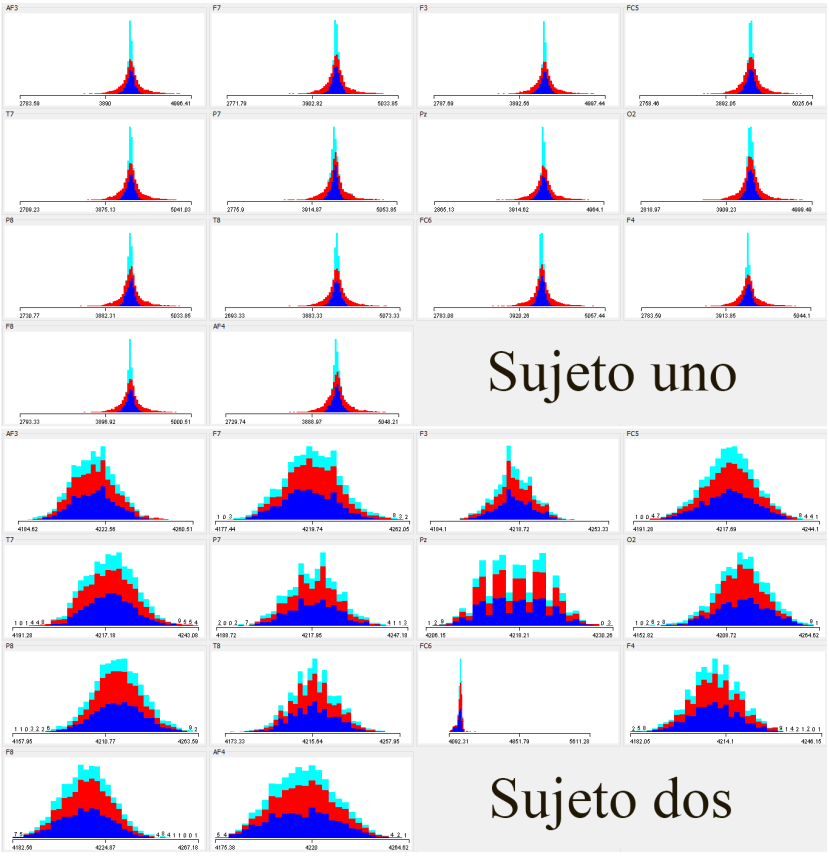

Figura 6.10: Comparativa de los datos entre sujetos

Como podemos observar la diferencia entre las formas de los datos que representan las ondas cerebrales es absolutamente distinta en ambos casos, por lo que se explica el motivo por el que el clasificador obtiene unos resultados tan bajos como para estar por debajo del porcentaje aleatorio a la hora de seleccionar la acción pensada por el sujeto.

6.5.3. Entrenamiento y clasificación con sets de datos de complejidad aumentada

Si decidiésemos utilizar 5 clases en lugar de tres, incluyendo los movimientos de izquierda y derecha asociados a las teclas “a” y “d” respectivamente. En este caso, la precisión desciende ligeramente debido a que cuantas más divisiones tenga que hacer un clasificador más incrementa la probabilidad de error del mismo.

Los resultados mostrados a continuación fueron obtenidos utilizando sets de datos tomados en sesiones distintas preprocesados aplicando el procedimiento de la media Laplaciana en grupos de datos.

Utilizando el clasificador Random Forest con los parámetros predeterminados obtenemos un 21,27 % de instancias correctamente clasificadas, frente al 78,73 % de instancias incorrectamente clasificadas, con una precisión media del 27,2 %.

Utilizando el clasificador KNN obtenemos un 21,51 % de instancias correctamente clasificadas, frente al 78,49 % de instancias incorrectamente clasificadas, con una precisión media del 25,8 %.

Como se observa, los resultados vuelven a niveles aleatorios, siendo mucho más bajos que los obtenidos por el mismo procedimiento con sets con tres clases de datos distintos, por lo que se descartó este tipo de procedimientos durante el proyecto.

6.5.4. Entrenamiento y clasificación con sets de datos tomados con los ojos cerrados

Como se explica en esta memoria, las señales captadas por los dispositivos de electroencefalografía son muy sensibles a las interferencias externas, el ruido y a las propias funciones del individuo del que se están tomando datos.

Una de las principales fuentes de interferencia en la señal son los propios parpadeos de la persona de la que están siendo captadas las ondas cerebrales, por lo que una manera de mejorar cualquier resultado con este tipo de dispositivo es grabar los datos manteniendo el sujeto los ojos cerrados.

Estas pruebas no son relevantes para el proyecto dado que se busca el movimiento del personaje en un videojuego por lo que es imprescindible la visión del usuario, pero se mencionará brevemente que se hicieron algunas pruebas referentes a este tipo de datos y los resultados mejoraban en la mayoría de los casos entre el 4 % y el 8 % las mismas pruebas con los ojos abiertos.

Capítulo 7

Resultados

En este capítulo se mostrarán todos los resultados obtenidos de manera que sea fácil visualizar el conjunto total de valores en tablas para los conjuntos de datos de diferentes sesiones y de la misma sesión.

7.1. Resultados con datos de sesiones distintas

A continuación se muestra una tabla con los resultados obtenidos en porcentaje de acierto de las clasificaciones para los datos clasificados utilizando un set de entrenamiento tomado en una sesión distinta.

Cuadro 7.1: Resultados para clasificación de datos en sesiones distintas

Clasificador VS Preprocesado	Datos en bruto	Criba	Criba selectiva	Media en grupos	Resta media	FFT
Random Forest	28,61 %	29,97 %	35,4 %	27,95 %	37,98 %	27,24 %
Random Forest modificado	31 %	31,01 %	35,01 %	29,39 %	37,55 %	25,4 %
Random Forest set grande	-	41,13 %	35,35 %	-	-	-
Random Forest set pequeño	-	-	-	42,65 %	**44,25 %**	-
KNN	**31,5 %**	31,13 %	**38,34 %**	38,7 %	40,42 %	**34,05 %**
KNN set grande	-	**42,2 %**	36,49 %	-	-	-
KNN set pequeño	-	-	-	**47,31 %**	39,67 %	-

Los Porcentajes resaltados en negrita representan el mejor resultado para cada tipo de preprocesado, mientras que el resultado subrayado representa el mejor resultado de la totalidad de las pruebas realizadas, siendo el mejor resultado obtenido hasta el momento el 47.31 % de instancias correctamente clasificadas.

Como se puede observar, prácticamente en la totalidad de los casos el mejor clasificador para este tipo de datos es el clasificador KNN, esto puede ser debido a las diferencias entre las señales, ya que habiendo sido tomadas durante sesiones distintas aparecerán mayores variaciones entre ellas.

Estos cambios pueden afectar en mayor medida a los clasificadores de tipo árbol que tienen a exagerar el ruido, haciendo que los porcentajes de acierto sean en algunos casos bastante menores que utilizando el clasificador KNN.

7.2. Resultados con datos de la misma sesión

En este apartado se muestra la tabla correspondiente a los resultados generados al clasificar sets de datos entrenados con otros sets de datos tomados durante la misma sesión sin que el sujeto se quite o mueva disposivo Epoc+.

Cuadro 7.2: Resultados para clasificación de datos de la misma sesión

Clasificador VS Preprocesado	Datos cribados	Media en grupos de datos	Transformada de Fourier
Random Forest	61,2 %	60,81 %	-
Random Forest modificado	**61,32 %**	**61,29 %**	-
Random Forest set grande	-	56,9 %	-
KNN	56,23 %	54,36 %	-
KNN set grande	55,9 %	57,46 %	-
Extra trees 10 sets de datos	-	-	83,98 %
Extra trees 8 sets de datos	-	-	**89,56 %**

Nuevamente, los porcentajes en negrita representan los mejores resultados para cada tipo de preprocesado y el resultado subrayado, el mejor resultado de todos los obtenidos entre las clasificaciones utilizando un set de entrenamiento de la misma sesión.

El motivo por el que se utilizó el clasificador "Extra Trees" fue que para este tipo de sets de datos, como se puede observar, los clasificadores en árbol nos ofrecían los mejores resultados con una diferencia de aproximadamente el 6 %, representando un cambio suficientemente grande como para querer indagar más en las posibilidades que podían ofrecernos.

Como podemos observar, gracias a la utilización de este nuevo clasifcador, los resultados obtenidos aumentaron su efectividad en un 37 % con el preprocesado basado en la transformada de Fourier, representando unos resultados equiparables a los mejores obtenidos con el propio software de Emotiv Systems.

Con estos resultados completaríamos todos los objetivos principales del proyecto, habiendo conseguido clasificar con unas de las mejores efectividades conseguidas hasta la fecha en cualquier tipo de pruebas relacionadas con este tipo de dispositivos low-cost.

Capítulo 8

Planificación, marco legal y presupuesto

8.1. Planificación

Todo este proyecto ha sido desarrollado durante diferentes etapas planificadas al principio del mismo. El periodo total del proyecto abarca desde Febrero de 2016 hasta Junio de 2016. Durante este periodo han ido surgiendo diferentes problemas para los que se ha tenido que encontrar algún tipo de solución antes de poder avanzar para lograr el objetivo final del proyecto.

La primera fase del proyecto duró desde mediados del mes de Febrero hasta principios del mes de Marzo, durante esta fase se establecieron las bases del proyecto, como se iba a plantear el desarrollo, el lenguaje de programación que se iba a utilizar, en cuyo caso Java fue seleccionado y el clasificador que mejor se adaptaba a nuestras necesidades, en este caso se eligió Weka. También empezó la curva de aprendizaje de Latex y Weka necesaria para la escritura de la memoria y la realización del proyecto.

Una vez sentadas las bases del proyecto comenzaría la segunda fase, esta se basaría en conseguir un enlace estable con el dispositivo Epoc+. Esta fase abarcaría desde principios de Marzo hasta mediados de Abril. Durante esta fase se plantearon dos objetivos. El primero fue conseguir obtener datos en tiempo real del casco Epoc+, esta parte tuvo algunos retrasos debido a que Emotiv dificulta mucho la comunicación siempre y cuando no se compre su software, como era el caso. Descubrimos que prácticamente la única forma de poder conseguir los datos del casco era "pirateándolo" de una forma no muy segura, por lo que al final se decidió que la mejor manera de avanzar en el proyecto era utilizar las librerías de Emotiv. Una vez programando con esas librerías en cuestión de un par de días tuvimos un enlace estable de datos en tiempo real entre el casco y el ordenador, guardando los datos deseados en un archivo .txt para su posterior procesado.

El segundo objetivo de esta fase fue la clasificación en tiempo real de los datos. Utilizando el clasificador Weka, se podía usar un set de entrenamiento de otra sesión o un set de datos tomado justo antes de la clasificación para entrenar el clasificador, una vez entrenado el programa comenzaría a tomar datos en tiempo real del sujeto clasificando las muestras correspondientes cada vez.

Durante todo el proceso de clasificación comienzan las pruebas en Weka de los diversos sets de datos y los distintos tipos de preprocesados para comparar y encontrar los mejores resultados aplicando así dicho procedimiento en la clasificación en tiempo real.

El objetivo de la tercera fase es conseguir una reacción del ordenador debido a la clasificación. Esta fase durará desde mediados de Abril hasta mediados de Mayo. En esta fase se busca que, mediante señales, el programa le diga al ordenador la tecla que debería pulsar según la clasificación obtenida. Una vez desarrollada la clasificación en tiempo real solo hay que añadir una pequeña parte a la programación para conseguir este resultado. Las primeras pruebas se realizaron con el bloc de notas, comprobando que efectivamente se pulsaban las teclas "w","s","a","d" o ninguna en el caso del pensamiento "neutral". Una vez esta fase estaba finalizando se pasó a realizar pruebas en videojuegos, consiguiendo mover al personaje del videojuego deseado al recibir el ordenador la orden pensada por el sujeto.

Fue necesario equilibrar la fluidez del movimiento del personaje en el videojuego ajustando el tiempo de pulsación de la tecla por cada orden clasificada para ayudar a la inmersión en el videojuego.

El principal problema del proyecto se dió cuando en la clasificación en tiempo real no funcionaba tan bien como se esperaba, las teclas pulsadas no correspondían con las acciones deseadas por el usuario. Debido a esto se precisó de un estudio intensivo de los sets de datos y la forma de procesarlos durante el mes de Mayo.

Este proceso comenzó probado distintos clasificadores de Weka con los datos en bruto obtenidos del casco Epoc+. Se decidió utilizar los clasificadores KNN y Random Forest tras estas pruebas.

El siguiente paso fue ir eliminando algunos de los atributos de los datos para comprobar hasta que punto se podían mejorar los datos sin dañar la información, se decidió que la mejor acción era eliminar los tres atributos que no aportaban informacion de la propia onda cerebral, "COUNTER", "INTERPOLATED" y "RAW_CQ".

Después de tener los menores datos posibles, se hicieron pruebas modificando los propios datos, las principales pruebas hicieron referencia a procesos relacionados con la media Laplaciana para eliminar la mayor cantidad de ruido, dejando de esta forma una señal lo mas nítida posible, y con la transformada de Fourier, intentando al cambiar de plano una mejor representación y lectura de la onda por parte del clasificador.

Tras todo el proceso de prueba y error, se llegó finalmente a la conclusión de que la mejor forma de operarlos es utilizar sets de datos de la misma sesión, utilizar la transformada de Fourier y entrenar el clasificador "Extra trees".

Tras conseguir unos resultados de clasificación tan buenos, el siguiente paso fue unir las tres partes, el proyecto actual, el preprocesado que mejor resultados daba y el clasificador que mejor se adaptaba a nuestras necesidades. De esta manera se esperaba conseguir un movimiento del personaje lo más acertado posible.

Durante todo el proceso se fue redactando la memoria según se hacían avances pero dado que no estaba terminada las semanas restantes se utilizaron para terminar de redactar la memoria, añadir los apartados restantes y terminar de arreglar el formato dejando el tiempo suficiente para su lectura por parte del tutor e impresión por parte del alumno.

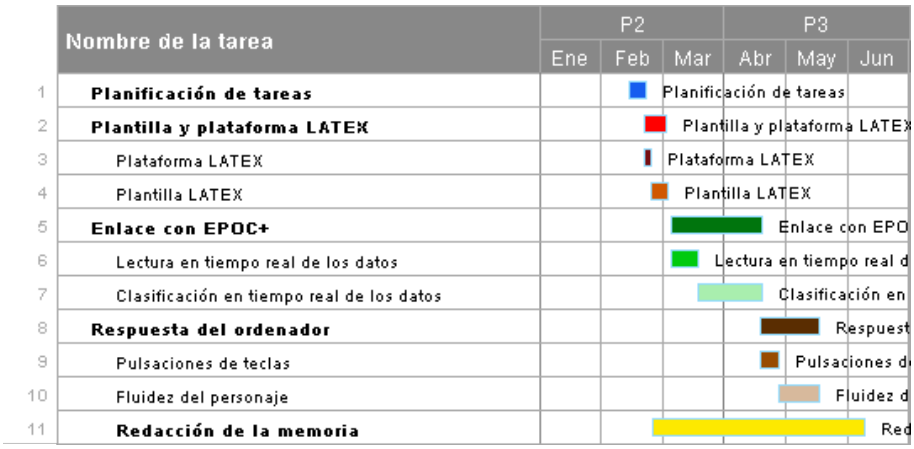

Figura 8.1: Diagrama de Gantt de la planificación

8.2. Marco legal

Cualquier programa desarrollado a partir de este proyecto no tendrá ninguna limitación legal más lejos de los términos y condiciones de uso de Emotiv Systems, los cuales son los siguientes:

- Acuerdo de licencia de usuario final:
  El uso del software y los servicios de Emotiv está sujeto a sus términos y condiciones junto a los términos y condiciones de EULA, en caso de conflicto entre ambos términos, prevalecerán los de Emotiv.

- Licencia:
  Emotiv concede una licencia limitada, no exclusiva, no sublicenciable, no transferible y revocable para instalar y usar el software en un PO compatible de la propiedad del usuario comprador para propósitos NO comerciales y privados.

- Derechos de propiedad:
  Emotiv conserva todos los derechos de software y servicio no concedidos específicamente en los términos de uso. Emotiv posee el software y servicios, incluyendo todos los derechos, título e intereses, incluyendo toda la propiedad intelectual y otros derechos de propiedad.

- Datos escaneados:
  El usuario conserva la propiedad de los datos escaneados con el sistema Epoc.

Aparte de dichas restricciones legales, se deberán tener en cuenta las restricciones legales referentes al uso del entorno virtual Eclipse, este software se encuentra bajo una licencia EPL (Eclipse Public License), cuyos términos y condiciones son los siguientes:

- Contribuciones:
  Cualquier contibución, ya sea modificación o añadido al código de Eclipse, serán considerados módulos externos al programa con su propia licencia.

- Concesión de derechos:
  Sujeto a los términos y condiciones, cualquier contribución concede una licencia "royalty-free" para la reproducción o desarrollo de dicha contribución. Si dicha contribución está patentada, dicha patente no hará efecto.

- Requerimientos:
  Todo contribuidor puede decidir distribuir su programa bajo su propia licencia siempre y cuando cumpla los términos y condiciones citados por Eclipse.

- Distribución comercial:
  Los distribuidores comerciales de software pueden aceptar ciertas responsabilidades con respectos a los usuarios finales y compañeros de negocios.

- No garantía:
  El programa se ofrece sin garantías de nigún tipo, expresadas o implícitas.

Para finalizar el último elemento de software utilizado fue Weka, este es software libre y está reconocido en la GNU General Public License. Utilizando una licencia GPL, siendo esta completamente abierta, Por lo que Su uso y modificación no tiene restricciones legales.

8.3. Entorno socioeconómico

El desarrollo de cualquier programa basado en este proyecto requiere que se tengan en cuenta los costes de los recursos físicos y laborales necesarios para la realización de la mismo.

Los recursos de hardware utilizados durante el desarrollo de este proyecto han sido los siguientes:

- Sistema Epoc+ de Emotiv Systems:
  En este paquete vienen incluido el casco Epoc+, los electrodos, solución salina para la conductividad de los electrodos, un USB que se encargará de la conexión inalámbrica del casco con el ordenador y el libro de instrucciones para que el usuario realice el uso correcto de todo el set. El precio del set ha sido de 799 euros.

- Solución salina:
  La solución salina es imprescindible para el uso del equipo Epoc+ y la cantidad incluida en el paquete de Emotiv es muy reducida por lo que ha sido necesaria la compra de más durante el proyecto. El precio por cada bote ronda los 3 euros.

- Ordenador:
  Por fortuna el software utilizado no requiere de un ordenador potente ni tampoco el programa desarrollado por lo que un ordenador de gama media debería ser suficiente. No obstante el ordenador que más se ha utilizado para el desarrollo del proyecto es de gama mediaalta, con un coste aproximado de 900 euros. Aún así los tiempos de entrenamiento del clasificador no han sido los mejores y en algunos casos el software de Weka se quedaba sin memoria por lo que sería aconsejable un ordenador de gama media con 16GB de memoria RAM.

Asimismo se han utilizado herramientas de software para poder desarrollar el programa y realizar las pruebas necesarias durante todo el proceso. Estas han sido las siguientes:

- IEDK de Emotiv Systems:
  Estas son las librerías necesarias para recibir datos del casco Epoc+, con la programación adecuada se encargarán de generar a través del enlace Bluetooth la conexión entre el casco y el ordenador,el envío de datos en tiempo real y realizar la desencriptación de los datos. El precio de este software es de 100 euros.

- Weka 3:
  Este es el software utilizado para la clasificación de los datos, ha sido necesario para realizar todas las pruebas de clasificación del proyecto. Gracias a él entrenaremos el clasificador deseado y se realizará la clasificación en tiempo real. Por fortuna para el desarrollo del proyecto este software es gratuito.

- Eclipse Mars 1:
  Este es el entorno virtual utilizado para el desarrollo del programa en lenguaje Java, facilitando la labor de desarrollo. Afortunadamente este software también es gratuito.

8.3.1. Coste total

A continuación se muestra una tabla con todos los costes asociados al proyecto, estos comprenden tanto el software como el hardware necesarios para la realización del mismo.

Cuadro 8.1: Costes totales del proyecto

Concepto	Precio en euros
Paquete Epoc+	799
Solución salina	3
IEDK	100
Ordenador	900
Total	1802

8.3.2. Estimación del precio del producto

Los costes de otros programas con la misma finalidad de éste proyecto rondan los 80 euros, pero teniendo en cuenta que este proyecto no está tan optimizado como dicho software, no tiene tantas funcionalidades, no ha habido un equipo grande de personas trabajando en su desarrollo y ha llevado cuatro meses de trabajo, su coste no debería ser mayor de 40 euros. De todas formas si alguien quisiera adquirirlo el producto se ofrecería bajo licencia open source y sería gratuito para la investigación sin buscar fines comerciales.

Capítulo 9

Conclusions

During this chapter, the conclusions made during this project will be analyzed taking into account the process and results obtained. Also future modifications and improvements will be included at the end.

9.1. Conclusions

The main goal of the project was to create a program able to classify in real time brain waves from a user using electroencephalography equipment based on the Emotiv Epoc+ headset, creating a connection between the headset and the computer, create the desired classifier, storing and pre-processing the training data and use it to train the classifier, and finally, store and pre-process the data to classify for the real time classification of that data.

As the real time classification is being made, each time a signal is sent to the computer telling it which action should perform, in the case of this project, keyboard keys signals would be sent so the movement of a character in a video game would be controlled.

As we have divided our work in several ways, the conclusions must be divided also, taking into account the same divisions, those would be, online and offline mode; inside offline mode, classification using a classifier trained with a set from the same or a different session and finally conclusions about the additional test performed.

9.1.1. Offline mode

This division of the work is related to all the test performed not in real time, this means, using Weka's GUI. During all this tests the main differentiation is made during the training, this could have been made with a set of data taken in the same session than the data to classify or from a different session.

Different sessions

As shown in chapter 7, we can see that this kind of training gives us a percentage of correct classification approximate to 50 % in the best case, with more data sets this could be improved but the time needed to train the classifier each time would be bigger and each set could also lower that value adding samples that do not represent the real desired though.

The best performance during this sort of test is given by KNN classifier, with a very little training set, this could be explained because, with data from different sessions, the position of the electrodes will be different, creating differences in the final signal, those differences were lower by using a Laplacian operator during the pre-processing but it seems that it was not enough for keeping the signals as similar as possible. Also with different sessions the thoughts can change in some way, the emotions of the users, the concentration level, etc. this will create even more differences between the sets of data making the classification even more difficult to be performed.

As Random Forest normally makes the noise, or differences between signals affect more than usual to the final classification results, this is the reason why only in one scenario during this tests Random Forest got better results than KNN classifier, we saw that, on the contrary, with same session data, tree classifiers got the best results every time.

Same sessions

During the test performed in this part of the project the results changed drastically, having even without operating the data 61,32 % of correctly classified instances, 14 % more than in the best case of training and classifying with sets from different sessions.

The results obtained sowed that tree classifiers did offer the best results for this sort of data using big training sets, this is why a new type of tree classifier was used trying to improve even more the results.

Those final results reached 89,56 % of correctly classified instances making the project reach the best levels of performance considered in the official software provided by Emotiv Systems.

We have to take into account that those results are obtained in the optimal way, and in a real time application could never be possible to obtain, neither with Emotiv software. There are too many variables affecting the signal, external interferences, low concentration level from the user, acoustic noises that changes the signal, etc.

As shown in chapter 7, those results were archieved using a pre-procesing of the data based on the Fourier transform, using the “Extra trees” classifier trained before classifieng the data.

9.1.2. Online mode

This part of the project represent the real time classification of data. This means select a previously taken training set, train the selected classifier with it and classify a real time stream of data taken from the user with the headset. As explained before, this creates the problem of bad classification using a training set from a different session. This can be solved taking the training set just before the real time classification starts but not every set we take are valid and this makes the process very long if it is needed to record several sessions for a large training sets every time; for the best results obtained during this project 1 hour was needed for recording the 10 sets of data.

Even with a valid set of data the classification depends on lots of external variables from the classifier and the training set, for example the user could move the headset changing the signal, the mental state of the person could change, giving the system false signatures inside the data that would lead to false positives during the classification...

Those are the main reasons why during the project a stable classification could not be reached at the end, because, even when using the official software provided by Emotiv it is very difficult to get such stable results.

9.1.3. Additional tests

Training and classification using only one set of data

During this type of test the classification could perform even the 98,35 % of samples correctly classified using the cross-validation method of Weka, this is the highest percentage obtained during the whole project. This is the least interesting aspect, but it could lead to other projects related to this one, for example prepare some sort of software which train a robot, or a virtual character previously for performing actions in a determined order. Those actions would be recorded by the user looking at the environment in which the actions will be made.

The good part of this kind of training is that it is only needed one set of data for training and classification, making the recording process very short and the computational force needed very low.

Training and classification using sets of data from different users

The tests performed in this part of the project were meant just for showing the reader that is nearly impossible to classify thoughts of one user using a training set of data recorded from a different user due to the extreme difference between both sets of data. This difference is given because each brain has a different shape and the neurons in the cortex are aligned in a different way, making the electric potentials go through the connections in a really different way from one user to another.

Training and classification using data sets with increased complexity

As the results did prove during the processing chapter, if we increase the number of actions that the classifier must differentiate, obviously the aleatory percentage decreases but also the percentage of correctly classified instances, this is due because for each new class to classify, the classifier needs a more precise range of values to select the correct output, making this objective more difficult to reach. Also with this kind of signals, the changes between one though and another are very little so it is needed an extremely precise system to perform such classification with a correctly pre-processed data.

In the main software from Emotiv we can observe that for three and four different thoughts the level of difficulty is called "hard" or "extreme".

Training and classification using sets of data recorded with eyes closed

This part of the project is just to prove how blinks affects the final results of the classification. As explained in the processing chapter, with the eyes closed, the classification process change the percentage of correctly classified instances in most of cases being between 4 % and 8 % better than results with the eyes open.

Obviously those results are not relevant as in the project the user will try to move the character in a video game and will need to see the environment, that is why those results are just mentioned briefly.

9.2. Future work

Having developed a program capable of recording and classifying in real time the thoughts of a user using electroencephalography technologies, there is room for many possible uses for such software and improvements related to the way of getting the training sets for the classifier. Also, with software developed using Java, there is always ways to improve the code for better performance.

In this section some of the possible modifications or improvements that have been coming to the mind of the writer during the whole project work will be enumerated and briefly explained.

- Unify program with game engine:
  As one of the objectives of the project is to move the character of a video game using Epoc+ headset, it would be useful to develop libraries that connect the headset with the game engine being able to change the options of the headset and making the process of recording easier.

- Design a video game based on the software developed:
  Now that we have the software for moving a character in a basic way, a simple video game could be developed in order to test in the best way possible the program as, being it created by the same programmer would deal to the aspects developed in the way needed.

- Record training data inside the video game:
  One of the main problems of the project have been the time needed for the record of the training sets, dividing it inside some parts of the game would make it less tedious for the user, making the experience much more entertaining.

- Change focus of the classification:
  Instead of move a character in a video game, this sort of technologies could make life much easier for people with disabilities, for example, if we could change the program so it could move a wheelchair instead of a virtual character could make a person with total paralysis due to tetraplegia move again using this device. If we talk about a person with certain mobility, we could still make his or her life easier if with this technology with actions as simple as open doors or turn on or off the light.

Apéndice A

Summary

During this append there will be explained an introduction to this project with the main objectives presented during the planning of itself. The implementation of the software is briefly explained. Also the results obtained at the end of the working period will be discussed, adding easy to read tables with all the obtained results. Finally the conclusions obtained at the end of the project are explained with some possible future lines of work.

A.1. Introduction

The brain is the most unknown organ in the human body, this is why the studies regarding this organ are considered of high value inside the scientific community. Because of that, during the last years a high number of studies related with this topic have emerged, trying to understand this organ and its functions more deeply.

Electroencephalography technologies are responsible of measure the electric activities generated in the brain of the patient using electrodes for the posterior analysis of the signal. The electric activity is organized in the most common and relevant brain waves for this sort of studies.

This project will focus on the study of the brain activity related to thoughts from the patient, classifying those thoughts using previously trained systems. The final goal of the project is to perform such classification in real time.

The point of view for the classification of thoughts will be centered around thinking about the movement of a virtual character inside a video game. The main difficulty of the project resides in the complexity of those kind of thoughts, as the person needs to think at the same time in the movement of the character, the action performed, the word and the key that is usually needed to be pressed in order to get the movement in the game.

A.2. Emotiv and Epoc

The hardware used for accomplish the objectives is based on a low-cost electroencephalography equipment from a company called Emotiv, this enterprise have more than one product related to such technologies but the one used during this project is Epoc+.

The main characteristic of this device given by Emotiv Systems are the following:

- Signaling:
  Number of channels:
  14 cchanels (references CMS/DRL, located in P3/P4)
  channels names:
  AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4

- Signal resolution:
  Sampling method:
  Sequential sampling.
  Sampling rate:
  128 SPS o 256 SPS* (2048 Hz internal)
  Resolution:
  14 bits 1 LSB = 0.51 micro Volts (16 bit ADC, 2 bits instrumental noise deleted), or 16 bits*
  Bandwidth:
  0.2 - 43Hz, digital Notch filters, 50Hz and 60Hz
  Filtering:
  Built in digital 5th order Sinc filter
  Dynamic range(referenced to the input):
  8400 micro Volts(pp)

- Connectivity:
  Wireless:
  2.4GHz band
  Bluetooth Smart

Emotiv also offers software for using and calibrate the equipment, users can find such software in a free version and also a version that needs payment. During this project the software used will be 'Emotiv Control Panel 2.0.0.21' in the free version.

This software have several tabs in it that mostly represent different programs, during this section the two most important tabs are commented, those are "Headset setup" tab and "Cognitive suite" tab.

- Headset setup:
  This tab will give us the relevant options for adjust the software for get the optimal results depending on the situation in which the data is being recorded, letting the user to be sure that everything is working as expected before the procedure starts.

  Every version in this software comes with a digital instruction manual which explains all the functionalities inside the software for its correct use, making the interaction with the program easier.

  The most important aspect of this tab resides in the image that appears in the left of the screen, in this image we can see the position of the electrodes in the head, this will guide the user for positioning correctly the electrodes for the right signal recording.

  All the electrodes in the image will change their color depending on how well they are positioned in the head of the patient. The color systems is the following:

  - Green: optimal signaling.
  - Yellow: good signaling.
  - Orange: good enough signaling.
  - Red: bad signaling.
  - Black: no signal

  Those states are calculated depending on the medium impedance in the electrode.

  In this tab the user can find also the battery and connection states and how many time have passed since the connection has been established.

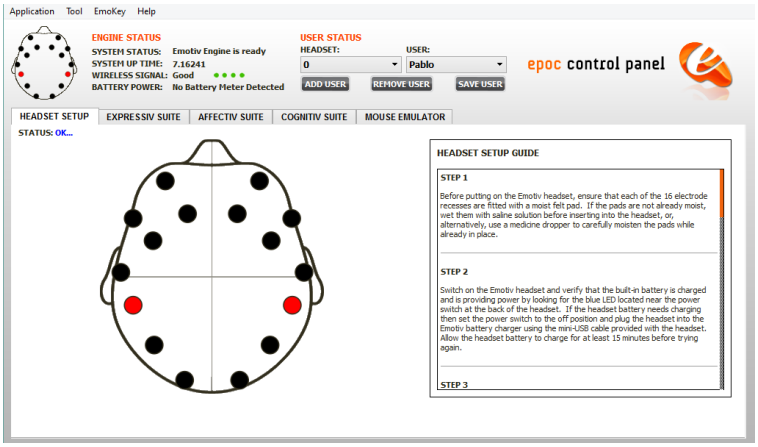

Figura A.1: “Headset setup” tab

- Cognitive suite:

  Inside this tab the user can find a multidimensional cube at the left with several actions at the right. This program will train a classifier in order to move the cube depending on the thoughts of the user in real time. A selection of four action for the cube can be made at the right of the screen as the classifier is not able to take decisions for more classes, also a state representing a calm state of mind in recorded.

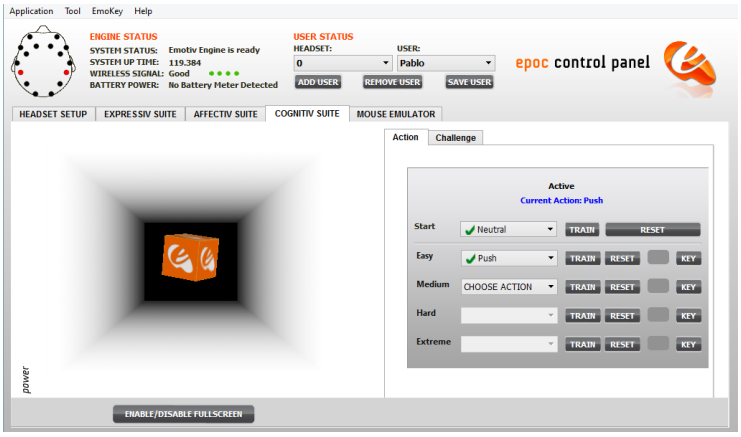

Figura A.2: "Cognitive suite" tab

A.3. Development

At the beginning of the project several ways for the development of it were proposed, as programing language, software and libraries to be used. The classification software selected at the end was Weka with the libraries needed for it. The programming languages proposed were the following:

- Java.
- C.
- Python.
- Matlab.

Finally, the selected language was Java and the virtual environment Eclipse Mars 1 would be used for making the process easier.

For using Epoc+ equipment Emotiv Systems IEDK libraries were needed with Java JNA and JRE libraries for the correct behavior of the software.

Once all the software and libraries are prepared, we have to start the process of development for the final program of the project. The part of this program must:

- Make the connection between the headset and the computer.
- Start the real time lecture of electric activity in the brain of the user.
- Record the data in a .txt file in the computer.
- Pre-process, if needed, such data before training the classifier and classify the data.
- Create and train the desired classifier.
- Classify in real time the data recorded from the brain activity.
- Send the needed signaling to the computer regarding the selected class during the classification.

A.4. Pre-processing

There has been selected several ways for pre-processing the data recorded in order to present such sets of information to the classification software Weka in different manners so it can interpret it depending on that selected pre-processing. Those are the following:

- Raw data:
  This is the most simple way to send the data to the classifier for training or classification, in this way the set is not modified, all the 17 attributes will be sent to Weka, including three of them than later on will be considered not useful. Obviously, this method gives us results that are lower than any other input already pre-processed.

- Not useful data deleted:
  In this kind of pre-processing the data will delete three attributes considered not useful, those will be "COUNTER", "INTERPOLATED" and "RAW_CQ" that will represent a count in the number of samples, the time stamp of the sample and the quality of the sample.

  During several tests it has been proven that those attributes reduce the effectiveness of the classification so deleting them is the best way of action, also reducing the time needed for training and classification.

- Selection of attributes:
  As deleting some of the attributes did increase the effectiveness of the classification, reducing even more the number of attributes could lead to an even better result. For this process a selection of seven over the fourteen total attributes will be made, the first, central and final seven attributes.

  The best results during this process appear selecting the first seven attributes but due to the complexity of this type of signals the number of such cases were not relevant for taking this method as valid for increasing the classification results.

- Laplacian mean:
  Using the Laplacian mean two different processes were developed, the first one will divide the sets of data in groups of ten samples and calculate the mean for each attribute, this will eliminate some noise and reduce the set making the time needed for the training and classification shorter.

  The second type will calculate the total mean from each attribute of each class inside a set and rest it to the value of all the attributes of the samples, centering all the means of the attributes around zero, this will eliminate even more noise but can lead to more loss of information.

- Fast Fourier Transform (FFT):
  This process will transform to the frequency domain all the data instead of having it in the time domain, this will give the classifier the most different way to interpret the data recorded, giving us a new chance to get better results during the training and classification.

  Using very big data sets for training the classifiers and using sets recorded in the same session, this method gave us the best results in the project, around 90 % of correct classification of samples, making such classification equal to the best results gotten with the official Emotiv software.

A.5. Results

For an easy to read results, a table is shown with all the percentages obtained during the tests made using sets of data for training recorded in a different session than the data used for classification.

Cuadro A.1: Results for classification of data from different sessions

Classifier VS Pre-processing	Raw data	Deletion	Attribute selection	Mean in groups	Rest mean	FFT
Random Forest	28,61 %	29,97 %	35,4 %	27,95 %	37,98 %	27,24 %
Random Forest modified	31 %	31,01 %	35,01 %	29,39 %	37,55 %	25,4 %
Random Forest big set	-	41,13 %	35,35 %	-	-	-
Random Forest small set	-	-	-	42,65 %	**44,25 %**	-
KNN	**31,5 %**	31,13 %	**38,34 %**	38,7 %	40,42 %	**34,05 %**
KNN big set	-	**42,2 %**	36,49 %	-	-	-
KNN small set	-	-	-	**47,31 %**	39,67 %	-

The percentages in bold represent the best result for the pre-procesing method, while the result underlined shows the best result of all the test performed.

As shown in the table, the best result is 47,31 % of the data correctly classified, this is an improvement of the 50 % over the aleatory selection of classes by the classifier during the classification process.

The other test performed are related with sets of data recorded during the same session, without the user moving or taking off the headset, this kind of data sets should keep more similarity between them, resulting in better results at the end.

Cuadro A.2: Resultados para clasificación de datos de la misma sesión

Classifier VS Pre-processing	Deletion	Mean in groups	Fourier Transform
Random Forest	61,2 %	60,81 %	-
Random Forest modified	**61,32 %**	**61,29 %**	-
Random Forest big set	-	56,9 %	-
KNN	56,23 %	54,36 %	-
KNN big set	55,9 %	57,46 %	-
Extra trees 10 data sets	-	-	83,98 %
Extra trees 8 data sets	-	-	**89,56 %**

The percentages in bold represent the best result for the pre-procesing method, while the result underlined shows the best result of all the test performed.

At the end of the project, the "Extra Trees" classifier was selected, as the best obtained results during this section of the project were obtained using tree classifiers, it was important to check all the possibilities offered from this new classifier based on this structure.

Thanks to the use of such classifier at the end of the project the levels of correct classification did reach 90 %, those levels represent the best levels reached using Emotiv software. With this kind of results, joining the rest of the parts of the project, the whole work can be considered finished.

A.6. Conclusions

The main objetive of the whole project was to develop a software program able to classify in real time thoughts from a user using an electroencephalography headset, the hardware used will belong to the company Emotiv and it is called Epoc+.

The program will create a connection between the headset and the computer, create the desired classifier, store and pre-process the training data and use it to train the classifier, and finally, store and pre-process the data to classify for the real time classification of that data.

While the real time classification of the data is being made, each time the classification is finished the program will send a signal to the computer representing the action to do, in the case of this project, keyboard keys signals will be sent representing the movement of a video game character.

The work has been divided in several ways, being those the following:

- Offline mode:
  Test performed using Weka's GUI. During all this tests the main differentiation is made during the training, being that division the following:

  - Different sessions:
    The best restuls obtained in this kind of tests are gotten by KNN classifier, using a small training set, the explanation for this is that, with data recorded in different sessions, the position of the electrodes could be different, creating differences during the data, the differences were lower using a Laplacian operator during the pre-processing phase but it was not enough for the desired results.

    Recording data in different sessions may lead to the thoughts to change, the emotions of the users, the concentration level, etc. may affect the data. This will create even more differences between the sets of data.

    Random Forest usually emphazises the noise, affecting more than usual to the final classification results, this is the reason why only in one scenario during this tests Random Forest got better results than KNN classifier, we saw that, on the contrary, with same session data, tree classifiers got the best results every time.
  - Same sessions:
    Without operating the data 61,32 % of correctly classified instances results were obtained, 14 % more than in the best case of training and classifying with sets from different sessions.

    Tree classifiers did offer the best results for this sort of data using big training sets.

    Results using those type of classifiers reached 89,56 % of correctly classified instances making the project reach the best levels of performance considered in the official software provided by Emotiv Systems.

    Fourier transform for pre-procesing, using the "Extra trees" classifier trained before classifieng the data was used for the best results.

- Online mode:
  Real time classification of data, selecting a previously recorded training set, train the selected classifier with it and classify a real time stream of data taken from the user with the headset.

Even with a valid set of data the classification depends on lots of external variables from the classifier and the training set, the user could move the headset, the mental state of the user could change...

Stable classification could not be reached at the end.

- Aditional tests:

  - Training and classification using only one set of data:
    Classification performed 98,35 % of samples correctly classified with cross-validation method of Weka, this is the highest percentage obtained during the whole project. Prepare some sort of software which train a robot, or a virtual character previously for performing actions in a determined order could be programmed.
  - Training and classification using sets of data from different users:
    Is nearly impossible to classify thoughts of one user using a training set of data recorded from a different user due to the extreme difference between both sets of data. The shape of the brain and the neurons in the cortex aligned in a different way, makes the electric potentials go through the connections in a different way from one user to another.
  - Training and classification using data sets with increased complexity:
    If we increase the number of actions that the classifier must differentiate the aleatory percentage decreases, decreasing also the correctly classified samples as more precise range of values is needed for the selection.
  - Training and classification using data sets recorded with eyes closed:
    blinks affects the final results of the classification. Changing the percentage of correctly classified instances to a range between 4 % and 8 % better.

A.7. Future work

Some of the possible modifications or improvements tought during the whole process of the project are the following:

- Unify program with game engine:
  develop libraries that connect the headset with a game engine being able to change the options of the headset and making the process of recording easier in case of it to be used for a video game.

- Design a video game based on the software developed:
  Develop a simple video game in order to test the program as, being it created by the same programmer would deal to the aspects developed in the way needed.

- Record training data inside the video game:
  dividing the time for recording the training sets inside some parts of the game would make the experience better for the user.

- Change focus of the classification:
  For people with disabilities, changing the program so it could move a wheelchair instead of a virtual character could make a person with total paralysis due to tetraplegia move again using this device. If we talk about a person with certain mobility, we could still make his or her life easier if with this technology with actions as simple as open doors or turn on or off the light.

Apéndice B

Electroencefalografía y el cerebro

B.1. Electroencefalograma

La electroencefalografía se basa en registrar la actividad bioeléctrica del cerebro. La actividad registrada evalúa los potenciales eléctricos de la actividad cerebral, esto se obtiene mediante una serie de electrodos que colocados en el cuero cabelludo o la base del cráneo del sujeto investigado[22].

La actividad eléctrica percibida por estos sensores será la actividad de las neuronas del encéfalo. Estas medidas se hacen basándose no solo en la magnitud en voltios que miden los electrodos, sino que también tiene en cuenta la frecuencia que estas señales detectadas portan. La actividad medida variará en gran medida dependiendo de la localización de los electrodos, entre otros factores como puede ser el momento en el que se toman las medidas, las diversas situaciones o los individuos. Esto implica que no es lo mismo tomar las medidas a un individuo que se encuentre en reposo que a ese mismo individuo realizando cualquier tipo de actividad.

Figura B.1: Ejemplo de un electroencefalograma

Existen otras maneras de captación de la actividad bioeléctrica cerebral. Uno de estos métodos se conseguiría mediante la colocación de electrodos en el cerebro expuesto o en localizaciones cerebrales profundas llamadas electrocorticograma y estéreo electroencefalograma respectivamente. Estos métodos son mucho más precisos debido a que los electrodos están en contacto directo con el cerebro y producen resultados muchos más fiables. No obstante, estos métodos son muy invasivos, lo cual supone un gran obstáculo ya que, para realizar un estudio sobre un número elevado de pacientes requeriría una cantidad importante de cirugías con un coste importante, conllevando esto un nivel de riesgo y peligrosidad importante y mermando la calidad de vida del paciente.

B.2. Señales EEG

Las ondas registradas durante la obtención de señales del electroencefalograma no suelen tener una forma de onda que las distinga entre ellas [24]. Pero ocasionalmente, esas ondas representan ritmos característicos que son divididos en Alfa, Beta, Theta y Delta:

- Ondas alfa:
  Presentan frecuencias comprendidas alrededor de los 8 y 13 Hercios, el voltaje que pueden presentar va desde los 20 hasta los 200 micro voltios. Lo normal es encontrar este tipo de ondas en personas despiertas con un estado mental relajado mientras mantienen los ojos cerrados aunque existe la posibilidad mediante estímulos externos de conseguir obtenerlas en pacientes con los ojos abiertos. Las zonas del cerebro en las cuales se registran este tipo de ondas son las regiones occipital y frontal.

- Ondas Beta:
  Presentan frecuencias comprendidas alrededor de los 14 y 30 Hercios, el voltaje que pueden presentar va desde los 5 hasta los 30 micro voltios. Dentro de estas ondas existen dos tipos claramente diferenciados llamadas Beta1 y Beta2, las primeras doblan en frecuencia a las segundas pero las ondas Beta2 únicamente aparecen durante una actividad importante del sistema nervioso central en momentos de estrés para el paciente. Las zonas del cerebro en las cuales se registran este tipo de ondas son las regiones frontal y parietal.

- Ondas Theta:
  Presentan frecuencias comprendidas alrededor de los 3.5 y 7.5 Hercios, el voltaje que pueden presentar es mayor de 20 micro voltios. Estas ondas se encuentran sobretodo en niños, aunque también son encontradas en personas adultas en situaciones de tensión o enfado, es posible encontrarlas durante la fase de Rapid Eye Movement (REM) del paciente. Se han realizado estudios que demuestran que este tipo de ondas están relacionadas con actividades creativas y con el subconsciente de los sujetos. Las zonas del cerebro en las cuales se registran este tipo de ondas son las regiones frontal y temporal [21].

- Ondas Delta:
  Presentan frecuencias comprendidas alrededor de los 1 y 3 Hercios, el voltaje que pueden presentar es inconstante. Este tipo de ondas son obtenidas durante las fases de sueño profundo del sujeto. La detección de este tipo de ondas es importante ya que si fuesen encontradas durante periodos en los que el paciente se encuentra despierto, podrían indicar problemas en el cerebro del sujeto.

Aparte de lo explicado anteriormente existen otras tres divisiones en estados según el voltaje y la frecuencia de la ondas. Manteniendo una frecuencia elevada pero un voltaje inferior encontraríamos un ritmo despierto. Por el contrario, si encontrásemos una frecuencia baja mientras el sujeto tuviese un voltaje elevado, ese sujeto se encontraría en un ritmo dormido. Por último si tanto la frecuencia como el voltaje fuesen al mismo tiempo elevados el sujeto se encontraría en un ritmo convulsivo.

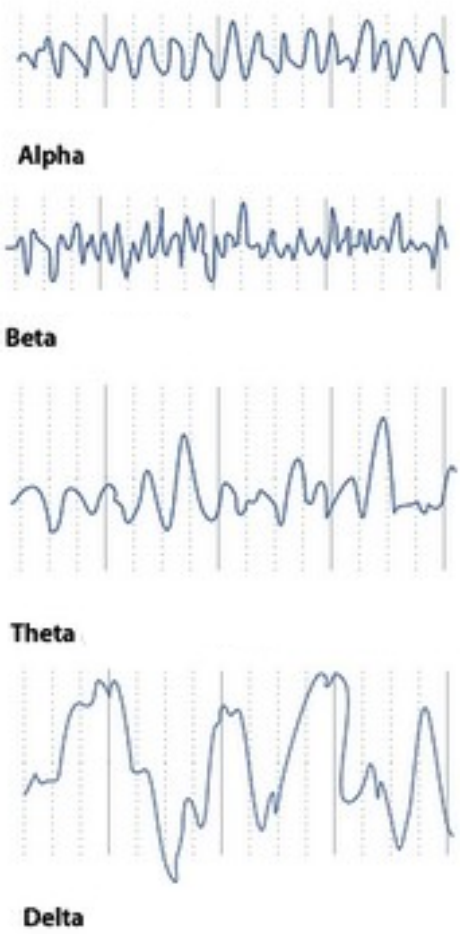

Figura B.2: Principales ondas cerebrales

B.3. Artefactos

Durante la recogida de datos de un electroencefalograma pueden aparecer artefactos, estos son distorsiones en lo que debería ser la señal real, estas distorsiones provienen de otras señales eléctricas no provenientes del cerebro que se suman a la que está siendo recogida. Estas señales dependiendo de donde provengan pueden dividirse en dos clases distintas. En la primera de las clases, esta actividad eléctrica proviene del propio sujeto y son denominados biológicos. La segunda clase de artefactos provienen de una fuente externa al sujeta y son denominados artefactos técnicos.

B.3.1. Artefactos biológicos

Los artefactos biológicos surgen de actividades inconscientes del paciente como por ejemplo respirar,lo normal es que este tipo de señales eléctricas presenten un voltaje mucho más elevado en los datos finales con respecto a la actividad cerebral normal durante el electroencefalograma. Las causas más normales para que aparezcan este tipo de artefactos pueden ser:

- Movimiento rápido de los ojos y los parpadeos del sujeto: Ambos movimientos generan pequeñas alteraciones en la zona frontal de la cabeza que influyen en las señales obtenidas.

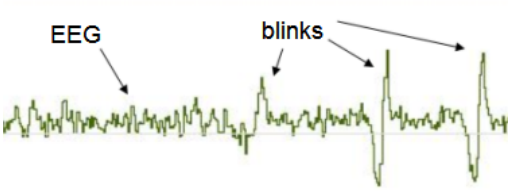

Figura B.3: Ejemplo de artefacto por el parpadeo del ojo

- Movimientos musculares: todos los movimientos musculares conscientes pasan antes por el cerebro, órdenes que se mandan a los músculos también serán captadas por los electrodos del casco de electroencefalografía alterando así la muestra obtenida.
- Movimientos musculares inconscientes: pese a que no son acciones pensadas por el sujeto, las órdenes que se les manda a músculos como el corazón también generan cierta actividad eléctrica que será recogida durante el electroencefalograma.

B.3.2. Artefactos técnicos

Los artefactos técnicos se refieren a cualquier alteración de la señal que dependa de la actividad externa al paciente, como cualquier campo electromagnético que provenga de las cercanás del sujeto, algunos de los causantes pueden ser:

- Red de electricidad: debido a que este tipo de pruebas necesitan de algún ordenador, necesitarán a su vez de una red eléctrica de la que este ordenador se alimente, en todas las casas, laboratorios y hospitales podemos encontrar este tipo de red que aunque no sea nociva para los usuarios si que puede afectar a materiales tan precisos como este tipo de cascos.

- Ruido térmico: ruido generado por los componentes del propio equipo de electroencefalografía.

- Electrodos: si en alguno de los electrodos utilizados para la obtención de la señal hubiera un cambio de la impedancia aparecería un artefacto en la señal en ese momento.

- Ambiente: Todo instrumento electrónico genera un campo electromagnético, por lo que las mejores condiciones para la toma de datos será con el menos número de este tipo de instrumentos cerca del sujeto y del casco de electroencefalografía.

B.4. Fisiología del encéfalo

El encéfalo se encuentra dentro del cráneo, es la parte más grande del Sistema Nervioso Central (SNC) y comprende el tallo cerebral, cerebelo y cerebro, a continuación explicados:

- Tallo cerebral:
  Es la parte del encéfalo que une el córtex cerebral, el cerebelo y la médula espinal. Se encarga de algunas funciones involuntarias como por ejemplo, que el organismo mantenga inconscientemente tanto el ritmo cardíaco como el respiratorio. Asimismo se encarga de algunos reflejos motores [18].

- Cerebelo:
  Se encarga de los movimientos voluntarios del sujeto, es decir, los movimientos conscientes, también se encarga de mantener el equilibrio del cuerpo mientras elimina todas las señales que podrían generar espasmos en los músculos de la persona.

- Cerebro:
  En el cerebro se localizan las funciones conscientes del sistema nervioso (sin incluir los movimientos voluntarios, de los cuales se encarga el cerebelo). Este está divido en dos hemisferios, donde cada uno se encarga de la parte opuesta del cuerpo, es decir, el hemisferio izquierdo se encarga de los miembros de la parte derecha del cuerpo y viceversa.

  La parte exterior del hemisferio se denomina córtex, y es donde se recibe la información sensorial. La corteza cerebral contiene 9 de los 12 billones de neuronas que contiene el cerebro humano en todo su conjunto, encontrándose estas en la parte más externa del cerebro.

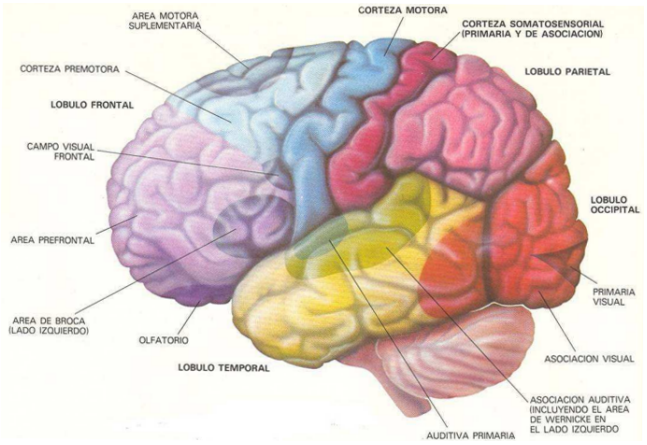

Figura B.4: Vista del encéfalo con sus principales lóbulos

En este trabajo nos centraremos en el estudio de las señales que se pueden obtener del cerebro, más concretamente de las señales eléctricas que aparecen en la corteza cerebral durante las sesiones en las que el sujeto deberá mantener diversos pensamientos en su mente.

Todas las señales sensoriales, pasan por la corteza cerebral donde se pueden encontrar ciertos patrones a distintas actividades. La zona de actividad relacionada con cada parte del cuerpo, es proporcional al número de nervios sensitivos de dicha zona.

La corteza cerebral tiene una forma tan irregular para maximizar la superficie de la misma, aumentando asi el número de conexiones neuronales. Las fisuras que dan lugar a esa forma son a su vez utilizadas para delimitar los lóbulos del cerebro, siendo estos:

- Lóbulo frotal:
  Es el principal encargado de controlar los impulsos, de generar el lenguaje, de acceder a la memoria, del movimiento del individuo, del estado de ánimo y las capacidades sociales. Toma parte importante en la programación y organización del comportamiento del individuo.
- Lóbulo parietal:
  Se ocupa de parte de los sentidos como podrían ser el tacto, la presión y la temperatura. También se sabe que es una parte del cerebro relacionada con el conocimiento matemático.
- Lóbulo occipital:
  Crea las imágenes que interpreta el cerebro, en él se encuentra la corteza visual, estando directamente relacionado con lo que el individuo ve.
- Lóbulo temporal:
  Realiza actividades relacionadas con la visión de manera compleja, como por ejemplo el reconocimiento facial.

También se encarga de obtener e interpretar la información procedente de los oídos, se ocupa del equilibrio del individuo y su coordinación. Tiene un papel muy importante a la hora de recibir e interpretar la información relacionada con los olores. Es el encargado de regularizar algunos sentimientos como la furia o la felicidad. Tiene relación con el lenguaje y el nombre que se le aplica a los objetos.

B.5. Unidad celular del sistema nervioso

La unidad más simple que podemos encontrar en el cerebro son las neuronas, es importante entender que debido a su composición estas células son las candidatas idóneas para enviar la información en forma de señales eléctricas a través del cerebro. Son capaces de percibir los estímulos y de transmitir los impulsos nerviosos entre células del mismo tipo e incluso a células diferentes.

Las neuronas están formadas por tres partes fundamentales:

- Soma:
  Constituye la parte más importante de la neurona. Dentro se encuentran subelementos como el núcleo, ribosomas o mitocondrias. Se encarga de metabolizar e integrar la información recibida ya que es precisamente por él por donde esta es transferida.

- Dendritas:
  Se extienden desde el soma en forma de ramificaciones, se podrían considerar enlaces de la neurona y se dedican a percibir estímulos al ser utilizados como los perceptores de las señales que provienen de otra neurona través de su axón.

- Axón:
  Es una extensión que parte desde el soma, tiene forma alargada, conduce la señal eléctrica hacia la siguiente neurona, normalmente tiene ramificaciones de manera que de una sola neurona puede llegar a varias a la vez.

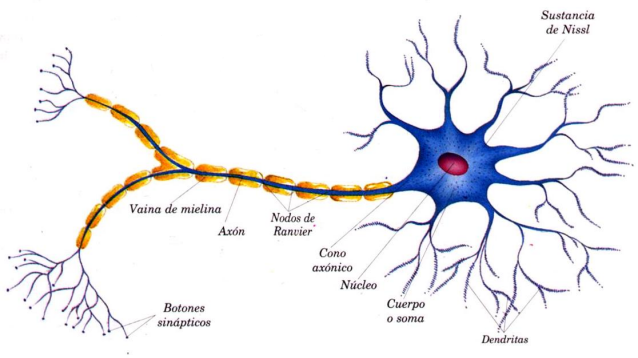

Figura B.5: Estructura de la neurona

Dependiendo de la manera en que las neuronas se unan entre ellas, estas estarán manteniendo un tipo distinto de sinapsis. Con esta unión las neuronas serán capaces de conseguir que la señal eléctrica siga avanzando por la red nerviosa hasta llegar a su destino. Hay dos tipos de sinapsis:

- Eléctrica:
  Se basa en el envío de iones de una neurona a otra utilizando acoplamientos de tipo gap, estos acoplamientos utilizan los canales generados por la conexión de complejos proteicos en neuronas fuertemente unidas.

- Química:
  En este tipo de sinapsis, la señal eléctrica comienza con una liberación de componentes químicos, la señal es generada en la membrana de la célula. Cuando esta señal llega al final del axón, esta célula libera ciertos neurotransmisores que se situarán en la zona entre la neurona originaria de la señal y la siguiente. Con los neurotransmisores la siguiente neurona sabrá si debe no no transmitir la señal eléctrica recibida.

Apéndice C

Obtención de potenciales eléctricos

Como se ha explicado anteriormente, las neuronas son capaces de enviar señales electricas a través de ellas, el principio básico para que los electrones se desplacen en un "cable" es la diferencia de potenciales eléctricos de los extremos del mismo. Si una neurona conduce o no dicha señal está generando esa diferencia de potencial en los extremos, al ser la neurona una unidad tan pequeña es imposible medir esa diferencia neurona a neurona. Actualmente la diferencia de potencial se mide en conjuntos de neuronas para medir ese valor [28].

El método utilizado para obtener señales eléctricas de una zona cerebral se basa en colocar electrodos en la base del cuero cabelludo, los electrodos recibirán diferencias entre los potenciales que reciben cada uno y esa será la información que envíen al equipo. Los valores que pueden tomar son tremendamente pequeños por lo que deberán ser amplificadas para ser más viable su observación [10].

C.1. Métodos para la obtención de potenciales según el número de electrodos utilizados

Obviamente, para calcular los distintos potenciales entre dos puntos necesitaremos al menos dos electrodos en distintas posiciones, esas posiciones no son aleatorias, existen zonas estudiadas que dan los mejores resultados para conseguir las señales deseadas. Una vez seleccionados los electrodos a utilizar y según los objetivos que se planteen a la hora de obtener y analizar los datos de la ondas cerebrales, existen varias formas de obtención de datos, estas son:

- Modo monopolar:
  Este procedimiento se realiza utilizando un solo electrodo para la toma del potencial eléctrico sin tener en cuenta el resto. Teniendo este primer potencial, el segundo se toma de un electrodo "de referencia" y la comparación se hace respecto al potencial de ese segundo electrodo, obteniendo así el resultado que será considerado el dato obtenido.

La situación ideal para este segundo electrodo sería presentando un potencial nulo, este caso es muy difćil de conseguir en situaciones reales por lo que se intentan conseguir potenciales muy próximos colocando este electrodo en lugares indicados para ello, como por ejemplo la barbilla del sujeto. Un sistema más complejo de conseguir una referencia funcional es aplicar a ese electrodo el sumatorio de potenciales obtenidos en el resto de electrodos que se le colocarían al sujeto excepto del electrodo con el que vamos a comparar el potencial. Lo normal es que ese sumatorio de cero, por lo que obtendríamos el resultado deseado sin necesidad de recurrir a aproximaciones. El principal problema que presenta este método es que como necesitamos utilizar todos los electrodos a la vez para poder realizar el sumatorio, solo será posible obtener la diferencia de potencial con respecto a un solo electrodo en cada toma de datos, alargando así el proceso por el cortocircuito presente entre el resto de electrodos.

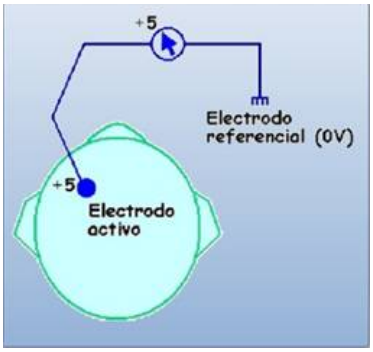

Figura C.1: Montaje y funcionamiento del sistema monopolar

La forma más conocida de esquivar ese aumento de tiempo en la toma de datos es utilizar resistencias con una impedancia baja para conectar todos los electrodos para tomar parte en el sumatorio, de esta forma se pueden tener tantos electrodos de referencia como se quiera, teniendo en cuenta los canales totales que se obtienen a la salida. Este método es conocido como el sitema Wilson.

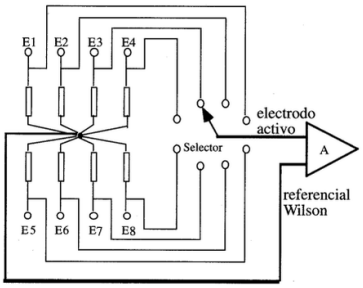

Figura C.2: Esquema del sistema Wilson para obtener datos en electroencefalografía

- Modo bipolar:
  Utilizando dos electrodos al mismo tiempo, se comparan los potenciales eléctricos de ambos, esta será la información que será mostrada en el equipo. De esta manera es posible conseguir muchísima informacion ya que habiendo normalmente un mínimo de 16 electrodos, el número de composiciones entre pares de electrodos es enorme. Hay que tener en cuenta que la mayoría de estas composiciones no nos aportarán información útil y que ya existen ciertas comparaciones óptimas para los resultados. Una ventaja de éste método es que al utilizar dos electrodos para cada obtención de datos se pueden generar a la vez varias tomas de datos ahorrando tiempo.

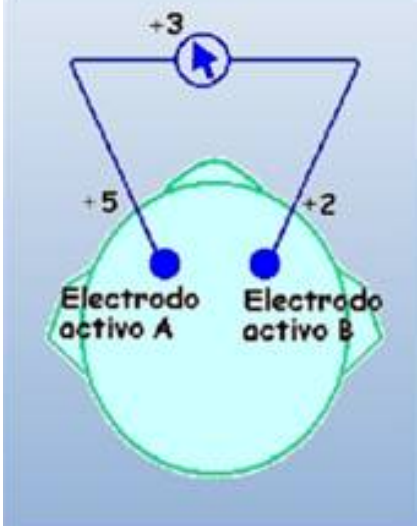

Figura C.3: Montaje y funcionamiento del sistema bipolar

C.2. Colocación de los electrodos en el cuero cabelludo

Hemos hablado de los modos de obtención de datos con respecto a la forma de utilización de los electodos, pero es importante tener en cuenta a su vez la posición de los mismos en el cuero cabelludo ya que dependiendo de la zona donde estén colocados obtendremos una señal u otra[26].

De todos los métodos conocidos, es más normal encontrar actualmente a la hora de observar un procedimiento de electroencefelografía el método diez-veinte. Ahora mismo este método es considerado el estandar de colocación de electrodos para la electroencefalografía.

Lo más usual en este método es encontrar la numeración de los electrodos de la mitad izquierda de la cabeza impar, siendo a su vez los electrodos de la mitad derecha de la cabeza números pares, los electrodos que se colocan en la separación de ambas mitades son conocidos como electrodos cero. Este es el sistema de nombramiento de electrodos americano.

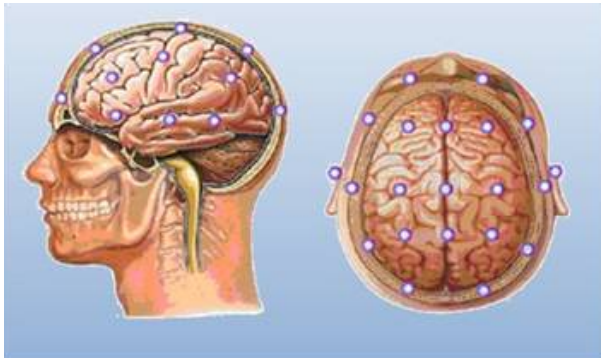

Figura C.4: Posicionamiento de los electrodos en el sistema 10-20

En el sistema europeo el nombre que reciben los electrodos varía un poco, junto con algunas posiciones ligeramente. De esta manera, al final tenemos los siguientes nombres para los electrodos:

- Temporales mediales (T3 y T4 en el sistema americano) pasan a ser llamados Tm.
- Temporales anteriores (F7 y F8 en el sistema americano) pasan a ser llamados Ta.
- Temporales posteriores (T5 y T6 en el sistema americano) pasan a ser llamados Tp.
- Frontales superiores (F3 y F4 en el sistema americano) pasan a ser llamados Fs.
- Parietales (P3 y P4 en el sistema americano) pasan a ser llamados P.
- Centrales (C3 y C4 en el sistema americano) pasan a ser llamados CI y CD respectivamente.
- Aparte aparecen un par de electrodos llamados mastoideos colocados en las apófisis mastoideas, son denominados por la letra M.

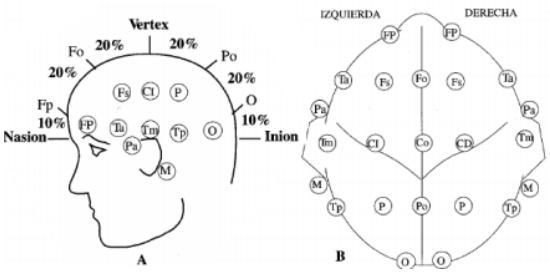

Figura C.5: Nomenclatura de los electrodos en el sistema 10-20 europeo

Es importante a la hora de la obtención de datos tener en cuenta lo siguiente:

- Para una configuración óptima serán necesarios dieciseis canales pero siempre y cuando haya un mínimo de ocho el resultado será fiable.
- El posicionamiento de los electrodos deberá seguir una configuración 10-20 para el mejor registro de datos posible[19].

Bibliografía

[1] Emotiv official web page. http://www.emotiv.com/.

[2] Emotiv, software development kit, user manual for release 2.0.0.20.

[3] Carolina Arboleda, Eliana García, Alejandro Posada, and Róbinson Torres. Diseño y construcción de un prototipo de interfaz cerebro-computador para facilitar la comunicación de personas con discapacidad motora.

[4] Michael R. Berthold. Guide to intelligent data analysis.

[5] Pavel Bobrov and Alexander Frolov. Brain-computer interface based on generation of visual images.

[6] Leo Breiman and Adele Cutler. Random forest - classification description.

[7] Daniel A. Driscoll and Stanley Rush. Eeg-electrode sensitivity- an application of recoprocity.

[8] Ehsan Tarkesh Esfahani and V. Sundararajan. Using brain-computer interfaces to detect human satisfaction in human-robot interaction.

[9] Sam Fok, Raphael Schwartz, Mark Wronkiewicz, Charles Holmes, Jessica Zhang, Nathan Brodell, and Thane Somers. Ipsihand: Direct recoupling of intention and movement.

[10] F.Ramos-Arguelles, G. Morales, S. Egozcue, R.M. Pabón, and M.T. Alonso. Técnicas básicas de electroencefalografía: principios y aplicaciones clínicas.

[11] S. Ha. Integrated circuits and electrode interfaces for noninvasive physiological monitoring.

[12] Tin Kam Ho. The random subspace method for constructing decision forests.

[13] J.L. Hodges. An important contribution to nonparametric discriminant analysis and density estimation.

[14] Tom V. Iancovici, Sebastian Osorio, and Jr. Bonie Rosario. Biofeedback in virtual reality applications and gaming, university of massachusetts lowell. introduction to biosensors. spring 2011.

[15] Eugene Kleinberg. An overtraining-resistant stochastic modeling method for pattern recognition.

[16] Juris Klonovs and Christoffer Kjeldgaard Petersen. Development of a mobile eeg-based feature extraction and classification system for biometric authentication, masters thesis, aalborg university copenhagen, june 2012.

[17] Gang Li and Wan-Young Chung. Estimation of eye closure degree using eeg sensors and its application in driver drowsiness detection.

[18] Iván Mora López-tercero. Detección de crisis epilépticas a partir de señales eeg mediante índices basados en el algoritmo de lempelziv.

[19] Falk Minow. International 10/20 system for electrode placement.

[20] Matthew K. Mukerjee. Neurophone: brain-mobile phone interface using a wireless eeg headset. paper presented at the proceedings of the second acm sigcomm workshop on networking, systems, and applications on mobile handhelds.

[21] T. Mullen. Analysis and visualization of theta-band information flow dynamics in an ern-producing task.

[22] Rafael Barea Navarro. Instrumentación biomédica apuntes de tema 5: electroencefalografía.

[23] Mavros Panagiotis, Coyne Richard, Roe Jennifer, and Aspinall Peter. Engaging the brain: Implications of mobile eeg for spatial representation.

[24] J.M Ramírez-Cortés, V Alarcón-Aquino, G. Rosas-Cholula, P.Gómez-Gil, , and J.Escamilla-Ambriosio. P-300 rhythm detection using anfis algorithm and wavelet feature extraction in eeg signals.

[25] F. Flórez Revuelta. Desarrollo de una interfaz cerebro-computador de bajo coste basada en máquinas de vector soporte.

[26] Homan RW., Herman J., and Purdy P. Cerebral location of international 10-20 system electrode placement.

[27] Mark R. Segal. Machine learning benchmarks and random forest regression.

[28] Andrés Castillo Silverio. Detección automática de paroxismos.

[29] B. D. VanVeen. Localization of brain electrical activity via linearly constrained minimum variance spatial filtering.